NBSIR 87-3566

Final NBS Report for CALS, FY86

Sharon J. Kemmerer, Editor

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Center for Programming Science and Technology
Institute for Computer Sciences and Technology
Gaithersburg, MD 20899

May 1987

U.S. DEPARTMENT OF COMMERCE
NA OF STANDARDS

NBSIR 87-3566

FINAL NBS REPORT FOR CALS, FY86

Sharon J. Kemmerer, Editor

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Center for Programming Science and Technology
Institute for Computer Sciences and Technology
Gaithersburg, MD 20899

May 1987

U.S. DEPARTMENT OF COMMERCE, Malcolm Baldrige, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

FINAL NBS REPORT FOR CALS

FY86

Table of Contents

TEXT

Specific Tasks

Textual Standards
Paper, SGML Parser Conformance Testing Methodology and Framework

DATABASE MANAGEMENT

Specific Tasks

Database Management Support
Paper, Preliminary Report on Data Management Standards

CONCEPT PAPER - CALS Representative Systems

EXECUTIVE SUMMARY

NATIONAL BUREAU OF STANDARDS
SUPPORT FOR
DoD COMPUTER AIDED LOGISTIC SUPPORT PROGRAM

FY-86 FINAL PROGRESS REPORT

EXECUTIVE SUMMARY
December 8, 1986

INTRODUCTION

The overall objective of the Department of Defense Computer Aided Logistic Support (CALS) Program is to integrate the design, manufacturing, and logistic functions through the efficient application of computer and communications technology. DoD requires functional and interface standards and procedures that will enable the digital interchange of data in weapon system and automated system contracts, that will be common to all Services and DLA.

Under an interagency agreement ratified in March 1986, the National Bureau of Standards (NBS) has provided assistance to DoD/OSD in support of the CALS Program. NBS identified four broad categories of standards required to support the interchange of CALS digitized technical information: (1) product data, (2) graphics, (3) text, and (4) database management. During the year, NBS activities associated with these four categories have been primarily aimed at:

1. Determining CALS requirements by familiarizing NBS technical staff with key DoD logistic functions and CALS demonstration projects:

   a. a two day DoD Logistic Seminar, presented by DoD organizations at NBS on 18 and 19 February 1986,

   b. travel to the sites of major logistic activities in Huntsville, Alabama; Dayton, Ohio; Los Angeles, Port Hueneme, Long Beach, and San Diego, California,

   c. review of many CALS documents such as the Service and DLA implementation plans, the CALS demonstration project descriptions, the report to Congress, etc., and

   d. a two-day workshop on Automation of Technical Publications & Engineering Data Repositories, held at NBS, which included presentations on key projects in DLA and the Services, along with NBS

staff presentations on current and emerging standards for data interchange.

2. Providing direct technology transfer by briefing DoD personnel, contractors, and other interested parties on federal, national, and international standardization efforts that are expected to support CALS objectives.

3. Identifying and recommending a preliminary set of standards required for data interchange in support of CALS.

4. Developing a plan for continuing support of the CALS program by assisting DoD in the effective use of relevant standards (enhancing, tailoring, determining performance, etc.)

FINDINGS AND RECOMMENDATIONS

The following recommendations, grouped into four general areas, identify actions that NBS believes will help to assure the success of the CALS development strategy recently published by DoD. Included are actions that lie within the technical expertise of NBS, and actions that NBS believes must be undertaken by DoD and industry cooperatively. Building upon the foundation laid during the past year, the CALS Program should during FY-87:

1. Extend the planning process to meet both near-term and long-term implementation requirements.

2. Document interactions among CALS functions and information processes to support implementation planning.

3. Develop core requirements, standards, and specifications to meet CALS application needs.

4. Establish the environment and tools needed to facilitate use of CALS standards and specifications, and to ensure the effective implementation of core requirements.

Specific findings and recommendations in each area are:

1. Extend the planning process.

Findings:

o The CALS development strategy provides an implementation concept, but a detailed implementation

process must still be articulated.

o Industry and government actions to develop and implement CALS technologies require close and continuing coordination.

Recommend FY-87 actions:

o The DoD and Industry CALS Steering Groups should clarify the organizational roles and relationships among industry, OSD, the Services, and DLA.

o A framework for development of CALS requirements and for phased implementation of CALS capabilities should be published.

o Industry should organize a consortium or other cooperative mechanism to focus on the major system integration and technology issues involved in defining and implementing the industry portions of Phase II of CALS.

o The NBS support program for FY-87 should focus on tasks which meet the priority requirements of DoD and industry for CALS Phase I core elements and which lay the foundation for Phase II core requirements.

o DoD and industry projects to develop, demonstrate, test, and prototype CALS capabilities should be effectively coordinated and given appropriate priorities in the DoD CALS Plan. DoD should establish procedures to prevent unnecessary duplication in CALS development efforts.

2. Document CALS function and process interactions.

Findings:

o The integrated design, manufacturing, and logistics environment of CALS requires rigorous definition of functional and information process interactions.

o Effective and efficient implementation of CALS requires a structured approach for selecting areas of development focus.

Recommend FY-87 actions:

o DoD and NBS should define a technology and standards baseline for near-term (Phase I) implementation of CALS capabilities.

o DoD should plan and implement a strategy to apply cost/benefit analysis, both at a corporate level and within each Service and DLA, to articulate the benefits of the CALS approach of standardizing interfaces between technical data systems, rather than the systems per se.

o DoD and NBS should develop, analyze and maintain a high level CALS data model as part of the overall CALS framework.

3. Develop core requirements, standards, and specifications.

Findings:

o Several industry standards (SGML, IGES, and CGM) provide capabilities needed to support CALS applications, but the potential role of many others examined during the past year is not yet well defined.

o The standards recommended for near term CALS use need selective enhancements, commonly agreed to implementations, and consistent application guidance.

o Interactions among standards, such as integration of document text and graphics, must be addressed as part of a comprehensive package designed to satisfy specified functional and user requirements.

Recommend FY-87 actions:

o NBS should continue analysis of CALS requirements, development of suitable standards, and development of implementation and application guidance to meet CALS needs.

o DoD should develop and validate near term (Phase I) CALS core requirements for highest leverage logistic application areas, incorporating recommended industry standards for digital exchange of data. In support of the Phase I core requirements, NBS should:

-- Develop DoD application guidance for SGML, IGES, and CGM, and incorporate it into an extended MIL-STD-1840.

-- Resolve issues regarding nonstandard implementation of digital exchange capability (e.g. CCITT Group 4 raster scan implementation).

o The Services and DLA should extend and tailor the initial Phase I core requirements and updates (Phase I.O., I.1, etc.), and undertake accelerated programs to test, demonstrate, and prototype these capabilities.

o DoD and industry should undertake a preliminary definition of long term (Phase II) core requirements to achieve the CALS functional and system integration requirements of the 1990s.

4. Establish the environment and tools for effective implementation.

Findings:

o Development of validation approaches is crucial to the successful implementation of the recommended CALS standards.

o High quality and timely availability of CALS capabilities, and good return on investment for developers and users are essential for CALS to succeed.

o Packaging, marketing, and two-way technology transfer are essential for industry and government acceptance of CALS technology and standards.

o Legal and contracting aspects of digital logistic technical information access and delivery are not fully defined.

Recommended FY-87 actions:

o DoD, NBS, and industry should jointly develop validation procedures and testing programs for CALS standards and for delivery of logistic technical information. As a starting point, NBS should provide a strategic plan for validation mechanisms for implementations of IGES, SGML, CGM and other recommended CALS standards.

o DoD and the CALS Industry Steering Group should resolve legal and contracting issues associated with the transition from paper-based to digital weapon system support processes.

PRODUCT DATA

I.1 PRODUCT DEFINITION DATA INTERCHANGE

SPECIFIC TASKS

FY 86

1. Assess DoD needs for Product Definition Data (PDD) interchange standards:

   a. Identify PDD requirements in terms of CALS applications (e.g. spares reprocurement)

   b. Recommend a set of PDD interchange standads for various product classes and logistic applications. Assess specific near and long term benefits, limitations and impediments to adopting these PDD interchange standards for DoD use.

   c. Assess specific current, intermediate, and long term capabilities to apply PDD interchange standards to competitive reprocurement of spare and repair parts for various product classes. Include an assessment of how such standards can be applied within legal competitive procurement constraints (DAR, FAR, Congressional direction, etc.).

   d. Identify and prioritize critical R&D issues in the development and application of PDD standards. Assess technical risks and provide rationale for the assigned priority.

   e. Develop a plan to expedite the development and implementation of PDD interchange standards for CALS based on the above findings.

Deliverables:

   -- Report to CALS Steering Group on tasks a-d (preliminary report 3 months after go-head, final report at 6 months)

   -- Plan for PDD area (outline 3 months after go-ahead, draft plan at 6 months, firm plan at 8 months)

2. In parallel with task I.1.1 assess the following specific PDD issues, and develop an issue paper on each:

   a. Potential advantages and disadvantages of IGES versus other graphics interchange standards (GKS, CGM, etc.) for CALS applications other than engineering drawings.

*b. PDD standards for electronics.

*c. Drawing scanning systems and standards.

d. Extent to which IGES technical issues (validation, flavoring) can be expected in PDES and other PDD interchange standards.

e. Approaches for supplementing near term and intermediate PDD standards with DoD specifications to ensure that product definition data contains, at a minimum, all the needed information currently provided in hard copy formats.

Deliverables:

-- Separate issues papers (incrementally delivered within 6 months after go-ahead).

3. Accelerate PDD standards development and validation efforts where needed to meet CALS schedule objectives:

a. Publish a comparison of industry/government IGES procurement practices and entity capability target dates.

*b. For IGES applications recommended based on task I.2.a, develop defined subsets of IGES entity types (this might include subsets for mechanical engineering drawings, electronic printed wiring assemblies, finite element mesh models, technical publications, etc.).

c. Assess priorities for future definition of IGES entity subsets, and provide a supporting rationale.

*d. Develop and publish a draft specification of data elements for labeling and identifying IGES files delivered to DoD.

e. Develop and coordinate Version 4.0 of IGES to include solids model data exchange capability. Develop supporting documentation and represent DoD interests to gain acceptance of IGES as an ANSI standard.

Deliverables:

-- Quarterly status reports and final technical report (6 months after go-ahead).

4. Assess IGES translators for interrelationships and interactions between entities for various combinations of CAD

systems to reduce the need for manual intervention or "flavored" translators:

a. Assess the extent of "flavoring" present in various vendor implementations and identify, with supporting rationale, which are correctable and which are not.

b. Review government (e.g. Sandia labs) and industry (e.g. General Motors) experience and assess the feasibility of obtaining 100% automated data transfer with IGES. Identify critical areas of needed work.

c. Assess alternative approaches to conducting validation of IGES translators. Recommend a DoD approach for an IGES translator certification program. Estimate the scope of such an effort.

*d. Develop and publish a set of guidelines for testing and validating IGES translators.

*e. Prototype a utility program to check conformance to a published subset of IGES entity types. Recommend an approach to develop a comprehensive utility program to check conformance of all approved entity types for specific applications, and estimate the scope of such an effort.

*f. Evaluate existing prototype programs to translate one specific vendor and version or level of IGES file to another "flavor" of vendor implementation. Recommend an approach for a comprehensive program to handle a greater range of vendor equipment and translators.

g. Prototype a utility program to check conformance to a specified data organization method.

Deliverables:

-- Quarterly status reports and a final technical report (8 months after go-ahead).

As DoD needs are determined, via the initial task,adjustments may have t made to the remaining tasks. Tasks identified by an asterisk (*) appear beyond the funding resources for FY86. These tasks will be accomplished in FY86 if possible. If not they will be deferred to FY87.

Tentative FY 87 and 88 Tasks

FY 87 and 88 tasks will be firmed up in the tactical plan delivered six months after FY 86 go-ahead. Tentative tasks

include:

1. Update of PDD standards plan.

2. Publish a specification on the use of layers for organizing the data in an IGES file.

3. Develop a program for automatically correcting incorrectly organized files and for generalized editing of data organization method.

4. Publish PDES Version 1.0 document.

5. Publish a working draft of an international standard for product data exchange (STEP).

6. Develop a utility program to convert IGES illustrations subset data into the Computer Graphics Metafile Format and publish an analysis of expected benefits.

7. Develop an Issue Paper on Configuration Control.

8. Publish Version 5.0 of IGES.

<u>FY88</u>

1. Publish analysis of error propagation after successive CAD data exchanges through IGES translators.

2. Develop validation techniques for solids model data exchanges and test cases to intentionally stress system limits.

DOD COMPUTER AIDED LOGISTICS PROGRAM

FINAL REPORT OF THE NATIONAL BUREAU OF STANDARDS
For Fiscal Year 1986 - March - September

PRODUCT DEFINITION DATA STANDARDS

The CALS program objective for digital product data is to have effective exchange of data throughout the life cycle of weapons systems development and deployment; an exchange using computer readable datasets describing the systems, their individual piece parts, and their product support data. A central issue here is the technology for digital representation of product data in its many forms: illustrations, drawings, 3D wire frame models, surfaced models, solids models, and complete product models.

Two terms will be used, product definition data and product data.

Product definition data (PDD) denotes the totality of data elements required to completely define the product. Product definition data includes the geometry, topology, relationships, tolerances, attributes, and features necessary to completely define a component part or an assembly of parts for the purposes of design, analysis, manufacture, test, or inspection. Very little if any process data is included, with the exception being conformance to a standard (MIL-STD) or reference to processes like a heat treat specification. The product definition is expected to be sufficiently complete as to enable the generation of all downstream process data.

Product data is more inclusive than product definition data. Product data includes all of the product definition data plus a larger class of data elements necessary to fully support the product for all applications over its expected life cycle. Product data is sometimes referred to as product model data.

The NBS CALS Program for product data exchange addresses the exchange, archiving, and future use by DOD of product model data. Major thrusts of this program are the development of a comprehensive program of testing and evaluation, the identification and solution of problems encountered in intersystem data exchange, research into the unique requirements for long term archiving, the development of software tools to assist users in making routine production use of digital product data, the continued development of new applications capability, the validation of new applications areas, and the acceleration of work directed toward making the use of complete product model data possible.

Throughout the DOD and its partners in industry, an increasingly larger number of computer aided design systems are being used in all phases of design, analysis, manufacture and test of weapons products. Over a hundred vendors offer these CAD systems. It is natural that different DOD activities or different companies would choose different vendor systems to meet their varying needs. Hence, there is a requirement in the normal course of business to be able to exchange the digital part models that are developed on one system to be used on another system.

Estimated at $4.3 billion in gross sales for 1986, the CAD industry is expanding quickly, and the capabilities of CAD systems are similarly changing. But the need for part model exchange among these systems has not diminished. Rather, with over 10,000 new CAD systems being sold each month, portability of data is even more important to the DOD and its contractors each day. The exchange of digital product models is expected to become as commonplace in the 1990's as the exchange of paper-based engineering drawings is today.

Since the databases of different vendor CAD systems are incompatible with one another, a direct transfer of digital product models is not possible. While converters from one database to another can be written, the only rational long-range solution on a company, national, or global scale lies in the development of neutral data exchange formats that are well documented, standardized, and implemented. Thus, the solution to these data exchange problems for both the DOD and the industry lies in the effective development of consensus approaches through close collaboration with standards-making groups like ANSI, and with independent groups like the Initial Graphics Exchange Specification (IGES) Organization, the Air Force Very High Speed Integrated Circuit (VHSIC) program, and the Electronic Design Interchange Format Organization.

It is essential for CALS to fully utilize the resources of these organizations to develop, review, and endorse needed pieces of the technology. Digital product data exchange cannot be developed by any one party. Nor can any of the parties afford the cost of a mistake in choosing the technical direction.

Work during FY86 in the area of product data exchange has centered on quantifying the present and the future needs of the DOD, identifying the problems with the use of IGES in the DOD environment, generating a series of technical issue papers, and developing a detailed plan of activities for FY87-88 that will assure an acceptable level of quality in IGES translators and diffuse the competence in data exchange technology throughout DOD.

The following sections represent work done toward Task 1.1 of the NBS statement of work in support of the DOD CALS initiative. Deliverables for this period include those for Subtasks 1.1.1.a b c d e, 1.1.2.a d e, 1.1.3.a b e, and 1.1.4.a b c g. Information contained in these sections is the result of NBS research, CALS workshops, formal IGES meetings, telephone interviews, and site visits to DOD installations at China Lake, Carderock, Huntsville, Dayton, Los Angeles, and San Diego. Included also is the experience gained through numerous discussions with DOD contractors on their successes and problems with digital product data exchange.

Task 1.1.1 Assess DOD Needs for Product Definition Data

The CALS team at NBS made several site visits to better understand how DOD and its contractors do business. Much interest in digital product data was evident, but only a few instances were noted of the exchange of digital product data:

DSREDS/EDCARS: A joint military program to store engineering drawings. System accepts CCITT Group 4 or paper input. Data is stored on optical disks in a compressed format. Plans include the input and output of IGES data, but do not include any capability to convert the IGES data to a form for reviewing or revising.

Pershing: An active user of digital product data. 60% of its engineering drawings are stored in IGES format. Drawings are maintained by the prime contractor who uses a system called MINGEL to manage and deflavor the IGES files. There are questions about access and control of the digital data. There is also an effort to convert existing drawings into IGES format.

ATOS: An important lead project for CALS. Tech Orders are entered into the system using SGML for the text and IGES for the technical illustrations. A special subset of IGES entities is defined for these illustrations and there is a parallel effort to incorporate these entities into a formal IGES subset. There is a critical need for a validation program to test vendor data for compliance to the IGES and SGML standards.

PDDI: The Product Definition Data Interface is an R&D project of the Air Force CIM Branch. Work has proceeded since late 1982 on the specification, pilot implementation and use of complete digital product data models. The work of the PDDI project is being integrated into the PDES efforts.

Rockwell B1-B project: The enormous size of this program has forced Rockwell and its subcontractors to automate their systems and integrate digital product data into them. Engineering drawings can be directly placed into Tech Orders and modified for display. They believe that their ability to deliver digital data packages significantly exceeds the government's ability to receive and use them. (EDCARS and ATOS are significant exceptions). One interesting point was that their efforts to automate caused criticism by a government inspection group because they were not following accepted procedures which presumably are established to deal with paper-based product data (drawings) and not digital datasets.

Subtask I.1.1a Identify PDD requirements

This element of the work plan is designed to identify DOD's needs for product data exchange and archiving. Applications areas are enumerated and characterized by their information content.

Logistics requires the ability to deal with digital product data for four generic applications:

1. Internal transfer of product data models among DOD components

2. The acquisition of new manufactured parts/systems

3. Data transfer to systems producing technical illustrations in documents referring to manufactured parts/systems

4. Archival storage of parts/assembly information

Requirement 1 - Internal Transfer

For many of the same reasons that engineering drawings are exchanged today, digital product data models will become prevalent in the near future. This use, however, is not easy to categorize since the range of applications is extremely large. Missile nose cone geometries, tank tread designs, footware sole pattern, machining geometries, technical illustrations and architectural floor plans each have their own requirements for data content and organization. Some applications, such as drawings, make use of simple modeling techniques such as wire frame geometry while more sophisticated applications such as tank vulnerability analyses require a solids modeling approach.

The impending Navy procurement of CAD workstations recognizes that five separate applications of CAD systems exist and that any one vendor system cannot effectively serve all applications.

Hence, the Navy strategy is to make multiple vendor awards and require IGES for internal transfer of product data among the dissimilar CAD systems; through IGES at first and then through PDES as it proves itself capable.

Numerous internal transfers of product models are found in R&D, prototype design, overhaul, and retrofit planning. Here the weapons system, assembly, or facilities design is analyzed in an "as built" configuration and modifications are devised and tested. Occasionally, a physical model is built in a prototype shop to test simulated performance. Designs, of course, must be reviewed by all levels of management. Frequently, small lot manufacture is done in house; as with submarine propellers at Philadelphia Naval Ship Yard or howitzers at Rock Island Arsenal.

Every one of these transfers of product information is a candidate for digital exchange in the immediate future. All that is necessary is to thoroughly test the applicable links and educate the appropriate personnel. While this is not a trivial task, it is finite, predictable, and easily implemented.

Requirement 2 - Acquisition of Parts or Systems

The second area of concern is the purchase by DOD of manufactured parts, assemblies, or whole systems. Recognizing that DOD maintains life cycle control over these purchased items, digital product data becomes an important consideration in the contractual relationship. Here the data follows the product from concept through detail design, engineering, manufacture, production, test, inspection, and deployment. The data package goes through repeated exchanges between primes, subs, government project managers, test labs, and consultants.

This external transfer application of digital product data is equally as large as internal transfer from the viewpoint of application areas and data content. But in dealing with dataset transfers in a contract situation, the problem is compounded with questions of data rights, liability, and dual authority.

While there may be many issues raised as to data rights, it is believed that the use of digital transfers introduces no change over the traditional use of engineering drawings. But invariably there will be contractors that say, "You can have my drawings but not my CAD data bases." As to liability, the issue becomes more cloudy. If the data exchange is perfect, it seems there is no more of a liability problem than when a drawing is furnished.

However, the question of liability for wrongful or incomplete information as a result of less than perfect software translators is difficult to prejudge.

Intriguing questions abound on the issue of dual authority - does the drawing or dataset take precedence? The formal engineering drawing of today is still the authority for product definition. Steeped in tradition, codified by ANSI standard, tested in the courts and cited by MIL-SPEC and MIL-STD, the drawing will remain a useful tool so long as man wishes to interpret the geometry, topology, tolerances, and features of a product design. Yet few will argue that someday we will place far more importance on the exchange of digital product descriptions than we do of paper. Drawings will become a by-product serving only to aid human understanding of the received dataset, not serving as a primary method for information exchange; however, no one is able to cite exactly when this will happen.

The term "dual authority" alludes to the interim period of time when datasets and drawings are both in prevalent use for data exchanges. Which dataset should take precedence if an inconsistency is found between them? Experience is that this happens often enough to require serious consideration. One instance was reported in which digital information in IGES format was in typical use. A contract definition of the prime authority was not provided. A last minute change to a critical dimension was made to the text note of a linear dimension in the CAD system, by the sender, rather than by modifying the part geometry that was being represented by the dimensioned drawing. Since the part was manufactured from the received geometry rather than from geometry created from the received CAD generated drawing, the part was wrong. Negotiations between sender and receiver in this instance placed the liability with the sender.

Most contracts still cite the drawing as the prime authority; however, the Boeing Commercial Airplane Company is taking the lead in citing the dataset as prime authority. While this is primarily done with families of parts of a non-critical nature, the application will most certainly spread as experience is gained. A booklet describing Boeing's contractual ground rules is reproduced in Appendix A.

Requirement 3 - Data Transfer for Technical Illustrations

Much work in this last year has addressed the resolution of problems with the digital exchange of technical documentation. The ATOS project in conjunction with the IGES Technical Publications Committee has developed and tested the section of MIL-STD 1840 which uses a subset of IGES for the exchange of the illustrations. This is quite appropriate since many of these illustrations are derived directly from the 3-D CAD product

model. The technique is recommended also by the Airlines Industry Association - Air Transport Association joint committee on Technical Publications, for exchange of documentation from the aircraft manufacturers to the commercial airline companies, e.g. between Boeing and United.

Requirement 4 - Product Data Archiving

If there is any one requirement unique to DOD, it is the need for archiving, or long term storage of product data. Present use and cumulative experience with digital product data has been in the area of exchange. Yet, like most government agencies, DOD has significant investments in data archives which are necessary to support its deployed forces. An almost unimaginable tens of millions of drawings are stored in data repositories dating back, for instance, to the recoil mechanism on the Civil War gun ship, Monitor. The problem as we approach storage of digital product datasets, is not particularly with the digital media or with the volume of information (although these do present challenges). Rather, the problem lies in satisfying the objective of being capable sometime in the future to achieve complete transfer of all information possible to the receiving system. This is hard enough to do presently when complete information can be obtained about the sending system. Archiving presupposes that the only information available is what is recorded on the retrieved file, and nothing can be assumed about the nature of the receiving CAD system of the future.

A ready solution is not at hand, and few companies have given serious thought to this problem. In characterizing the information content of a digital product file for archiving, it would be safe to theorize that in addition to the entity content of the exchange file in the application area specified, additional data might also be included to record how the application information was mapped into the IGES entities. But this is still an area of research.

Requirement 5 - Transition from Paper to Digital

There is also a need to allow vendors who do not have modern CAD systems to be able to do business with DOD. Essentially this means the government must continue to deal with blueprints and aperture cards for the foreseeable future. At a minimum, the government contracting process should make provision for the use of data centers where drawings could be converted into digital datasets or datasets into drawings. It is expected that the need for paper-based product data will steadily decrease as an integrated digital environment becomes established.

Subtask 1.1.1.b Recommend and Assess PDD Standards

This element of the work plan is intended to recommend a set of PDD interchange standards for various product classes and logistic applications, and to assess specific near and long term benefits, limitations, and impediments to adopting these PDD interchange standards for DOD use.

There are currently a number of national efforts to develop product data standards. In the area of mechanical products there are two; the IGES data exchange specification and the PDES product data exchange specification. IGES has also addressed the area of printed circuit board products. Two efforts exist in the area of integrated circuit products; the Electronic Design Interchange Format (EDIF) and the Air Force sponsored VHSIC Hardware Definition Language (VHDL).

These evolving standards, IGES, PDES, EDIF, and VHDL should be the only standards efforts adopted by DOD. Others such as APT, IPC D350-2, and COMPACT are good standards for their area of application, but are not product representation formats. Primarily they address the process requirements to generate control data for automation equipment. As such, they are can be derived directly from the product data formats or with the help of manufacturing engineers and part programmers. Their use varies with the production shop.

Any meaningful evaluation of these product data standard efforts must take into account several facets of a successful work effort;

1. application area
2. maturity
3. standardization status
4. degree of implementation
5. resource commitment

While most of these classifications are self-explanatory, maturity and resource commitment will be elaborated upon. Maturity means a measure of completeness and technical worth as proven by time, and industry consensus. Resource commitment has to do with the number of people that stand in back of a specification to fix it if necessary, or to extend it to new capabilities. Commitment also addresses whether the group has a long term pledge to stick with the area or will be disbanding as soon as the paper is published or the funding runs out.

Area of Application

IGES and PDES have primary application to mechanical, finite element, electrical PCB, and AEC models. EDIF and VHDL refer to IC products. Certain overlaps are noted in IC and PCB between EDIF and IGES, but NBS is working toward a near-term resolution of these problems.

Standardization Status

The only specification for product data exchange currently accepted as a national standard is IGES.

IGES Version 3.0 is currently being reviewed by ASME Committee Y14.26 as a new ANSI Standard with an expected approval in the third quarter of 1987. A thorough discussion of IGES is given under Subtask I.1.2.a.

PDES is expected to be available in preliminary review form by the second quarter of 1987 and in a working draft by the fourth quarter of 1987. After being passed by the technical committees, it will be submitted to ASME Y14.26 for national standardization and to ISO TC184/SC4 for international consideration. A thorough discussion of PDES is given under Subtask I.1.2.a.

Presently, the EDIF committee is aligning itself with the IEEE Data Automation Standards Committee, the ASME Y14.26 and with the Electronic Industries Association. No timetable is known for achieving a national standard on EDIF 2.0.

Maturity

IGES started in 1979. Version 1.0 was published in early 1980 with Version 2.0 available in 1983. Version 3.0 came out in April 1986 and Version 4.0 is scheduled in the second quarter of 1987. Version 1.0 was approved as an ANSI Standard in 1981 and Version 2.0 was evaluated favorably under a $500K Air Force contract in 1982-3. All improvements suggested in these reviews have been coordinated into the specification and are reflected in the present version. IGES is production worthy and useful for transferring part geometry and attributes between different CAD systems. Though IGES has known problems, primarily with the quality of vendor translators, it is being used in production at a large number of companies.

PDES is still in the R&D stage, but offers the promise of being able to describe the complete set of information needed to manufacture a part. PDES is not meant as a replacement for IGES, it is seen more as an advanced technology. It must be recognized, however, that a workable implemented PDES capability

is still a couple of years away from quality vendor implementation.

The EDIF activity began in November 1983 and is finishing Version 2.0 which is to be released in November 1986. The EDIF organization, like IGES, numbers around 100 quite active contributors, with a larger group (around 600) semi-active.

Degree of Implementation

IGES has been implemented for at least two years on every major CAD system marketed in the U.S. with over a 1% market share by gross sales. EDIF implementations are available on CADNETICS and VALIDLOGIC.

Subtask I.1.1.c Assess Procurement Applications

This element of the work plan is designed to assess specific current, intermediate, and long-term capabilities to apply PDD interchange standards to competitive reprocurement of spare and repair parts for various product classes.

To make use of digital product data in reprocurement actions, the requirements for a neutral, archival quality dataset must first be understood, and then the digital data itself must be made available. The first step, understanding the requirements of an archival quality dataset, requires more experimentation and some R&D addressing extent to which different flavorings presently exist among CAD vendors, and developing mechanisms for minimizing their influence. The second step, making the data available for reprocurement actions, involves procuring the right data with each new contract, generating the data at optimal cost on spares contracts, or generating the data in-house during a redesign or remanufacture activity.

What is needed are specifications for each application area subset of IGES so that datasets can be procured and reissued as part of orders for spares procurement with the confidence that the data can be successfully utilized. The problem of dual authority and the need to establish the precedence of datasets over drawings was discussed earlier. It is recomended that the furnishing of IGES data as part of contracts for spares production should initiate with the IGES datasets being provided as information only, until such time as digital data is found to be unquestionably reliable.

These application area subset specifications must be developed specifically for the application being addressed in the data exchange. Procurement specifications have been developed with this principle in mind by the Navy Sea Systems Command, by General Motors, and by Hughes EDSG. The procurement

specifications prepared by GM and Hughes are given in Appendices B and C.

The question remains as to how the IGES datasets will be generated. Two avenues are possible:

1. An optional specification can be attached to each procurement of spare parts. This spec would say that if the contractor uses CAD in the contract, the government offers to purchase the IGES file for an extra fee provided it passes certain acceptance tests.

2. The contract for spares could be processed in-house by a government facility, provided they use CAD to model the part and agree to furnish the IGES file along with the finished parts. As above, the IGES file must pass certain acceptance tests.

Both the procurement of product data as IGES files and the testing of translators requires an appreciation of application subsets. Each new application area in IGES will require a definition of information content through formal modeling techniques, a specification of how this information is unambiguously mapped into or represented by the IGES entities, and a definition of the IGES entity subset for the application. Also needed will be a series of test cases, a method for organizing the data within the IGES file, and demonstrations of intersystem exchange. Already several of these subsets are being prepared by committees of the IGES organization or by private companies. The Electrical Applications Guide is one example in the PC board area and MIL-STD 1840 is another in the technical illustrations area.

In February, the Navy circulated a draft Request for Procurement on new CAD systems that reflects their intent to make frequent use of digital product data. A detailed, mandatory IGES requirement is included. NBS personnel have worked with the Navy to refine these specifications and develop needed verification procedures.

Legal issues were investigated through a review of the 1985 CALS panel report and through collaboration with the US Navy CAD procurement effort, with the DOD MIL-STD Standard 1840 project and with Hughes and General Motors on the development of uniform CAD IGES procurement specifications.

Subtask I.1.1.d Identify and Assess Critical R&D Issues

This element of the work plan is designed to identify and assign priorities to critical R&D issues in the development and

application of PDD standards. It also assesses technical risks and provides rationale for the assigned priority.

IGES Testing

Several potential areas of R&D are associated with the development of IGES testing methodology. The work would lead to the development of a rigorous set of software utilities to assist with the validation of translators and procured datasets. Appendix D contains the details of one proposal to NBS for this work. Work is also needed in the application of artificial intelligence techniques to increase the degree to which the results of formal testing activities are automatically interpreted. Current practices require the expenditure of considerable time in the manual review and comparison of data listings in order to accomplish the evaluation.

IGES Data Archiving

The experience to date in using digital product data has been reasonably successful. Yet this has only been the result of careful planning and testing of the data paths used. The effort has been slow because of the need for personnel involved to understand the idiosyncracies of the CAD systems and to develop competence in using the translators. Success can be attributed to the collaboration of sender and receiver. This is a reflection of present practice world-wide and underscores that the present use of digital product data is for exchange.

The term "exchange" denotes an activity characterized by the following conditions:

- The data transfer occurs at a single point in time
- Complete information is available on the two computing systems involved
- Sender and receiver can communicate to solve any problems that are encountered

However, the problems envisioned in the DOD CALS arena include not only "exchange" but also "archiving" of product data. Archiving is a much more demanding activity. Archiving is characterized by:

- Data transfer to a receiving system at some relatively unknown future point in time

- Only limited information known about the CAD system and the computing environment which prepared the product data

- Complete information available only on the receiving system

- Receiver acting alone must resolve any problems that are encountered

- Generator of product data files has no real vested interest in assuring that data is complete and accurate (Future use of the data may not be the same as when the data was prepared. For example, data produced for human consumption as an engineering drawing may be needed in the future for NC data generation.)

As can be seen, the archiving problem is more severe than the immediate exchange of data. In order to make archiving achievable, detailed guidelines and software tools are needed to analyze all datasets and assure their data integrity before they are accepted and placed into archival storage.

Subtask 1.1.1.e Develop Plan to Expedite Standards Development

This element of the statement of work is designed to develop a plan to expedite the development and implementation of PDD interchange standards for CALS, based on the above findings.

A plan to expedite the development of product data standards is outlined in figure 1 and more fully presented in Appendix E. The plan is an expanded version of the plan developed by the IGES Committee to guide the development of both IGES Version 4.0 and PDES Version 1.0. That document constitutes a formal long-range plan for the IGES Organization, approved by the IGES Steering Committee on March 26, 1986. The plan presented here also reflects an increased emphasis on testing of translators, development of utility software for procurement, and an acceleration of test case development; all of which are critical to the CALS plans for the successful implementation of digital product data. Schedules and target dates cannot be forecast until resources under the CALS program are finalized.

Task 1.1.2 Technical Issue Papers on Product Data

This element of the work plan is designed to develop several technical issue papers having to do with IGES versus other standards, exchange completeness, and translator testing.

I. Develop Comprehensive Testing Methodologies

II. Develop Entity Test Cases

III. Develop Applications Area Subsets

IV. Develop Production Worthy Translators

V. Perform Extensive Intersystem Testing

VI. Implement a Translator Validation Program

VII. Accelerate PDES Development

VIII. Pursue International Standards Approval and Liaison

Figure 1. A Plan to Expedite the Development and Implementation of Product Data Definition Interface Standards.

Two tutorials are presented in the Appendices. The first, in Appendix F, is on the completeness of product data exchange. It examines the hierarchy of information flow from the application on one CAD system through the IGES entity exchange and the media or telecommunications system to the receiving CAD system. The second, in Appendix G, has to do with testing methodology. Developed by the IGES Testing Methodology Committee, the document is in working form at the present time and is expected to be approved and published in 1987.

Task I.1.3 Accelerate Product Data Development & Validation

This element of the work plan is designed to accelerate the development of standards for product data exchange as well as the development of validation techniques as needed to meet CALS schedule objectives.

The Product Data Exchange Specification is being developed as an exchange mechanism for complete product models. The PDES Initiation Activities were designed to develop the concepts and the methodologies to be used in PDES and to rigorously establish PDES information content as a baseline for Version 1.0 development. A report on the details of the presentations has been prepared and distributed. Much additional work is needed to bring the PDES effort to the point of a Version 1.0 specification and then to validate the concepts and implement sample code. The Air Force Advanced Tactical Fighter (ATF) program and the Geometric Modeling Applications Program (GMAP) both have reviewed the PDES material from the April 1986 Initiation Activities and agree that the program should be accelerated. The NBS Automated Manufacturing Research Facility could provide a responsive testbed for such work.

The Plan in Appendix E addresses several avenues for accelerating the work in product data standardization. It presently shows little emphasis on PDES, as the CALS direction is perceived as having heavy reliance only on IGES for the near future.

Task I.1.4 Iges Translator Capability Improvement

The context of product data exchange in the DOD environment is so broad that ensuring a good quality of exchange through the neutral IGES format requires careful specification of the data requirements for each application, complete testing of the software involved in the exchange, development of software to assess compliance at critical points in the exchange, and extensive intersystem testing to identify and evaluate problem areas, maintain area competence, and provide needed feedback to CAD system developers.

Copies of Appendix A, Boeing/Supplier CAD/CAM Interface Standards, April 1984, can be obtained from:

The Boeing Company
P.O. Box 3707
Seattle, WA 98124

APPENDIX B
****

GM/IGES Specification

December 22, 1983

Computer Integrated Systems
General Motors Technical Center
Warren, MI 48090-9010

Document Number CGS 101-F-01

GM/IGES Specification (CGS 101-F-01)

First Edition published December 22, 1983.

Revisions distributed March 30, 1984.

Refer to memos kept at the back of the manual for details on the the material that was revised.

ABSTRACT

In conjunction with NBS/IGES Version 2.0, this document provides the specification for CAD/CAM vendors' compliance with General Motor's required Initial Graphics Exchange Specification (IGES) standards. Specifications for data form, geometry, and structure entities are included. Entities that are not required at this time are also identified in this specification. GM's vendors are expected to support the entities identified in this document through their IGES translators.

PREFACE

This specification will be updated regularly until GM achieves 100 percent data exchange. This document refines the IGES specification to meet GM's
* needs. It has been reviewed and approved by the GM Data Exchange Committee,
* which is the Corporate team responsible for the implementation of data
* exchange specifications within GM (see "Appendix A. Members of the GM Data
* Exchange Committee" on page 5.1). This specification will evolve as technology and GM requirements change. Future releases of this document will incorporate adjustments due to the release of new versions of the NBS/IGES standard, as well as additional needs of GM for electrical, numerical control, finite element modeling and analysis (FEM/FEA), and solid modeling
* applications. Ongoing maintenance of this specification is the responsi-
* bility of the GM Data Exchange Committee and the Extensions and Repairs
* Subcommittee (see "Appendix B. Members of the GM Extensions and Repairs Sub-
* committee" on page 6.1).

* The authors of this specification assume that the reader is familiar with the technical aspects of computer graphics and NBS/IGES Version 2.0. Unless stated otherwise, any reference to NBS/IGES within the document refers to NBS/IGES Version 2.0, published in February, 1983.[1]

This document is organized to resemble the NBS/IGES Version 2.0 document. Unless stated otherwise, the definition of terms in this specification is identical to that used and defined in NBS/IGES Version 2.0. The Directory Data Entity Type Number of each entity described in this document refers to
* the number listed in the NBS/IGES Version 2.0 document. To obtain a copy of
* the NBS/IGES Version 2.0, write to the following address.
*
* National Technical Information Service
* 5285 Port Royal Road
* Springfield, Virginia 22161 Order Number: PB 83-137448
* For further information, phone (703) 487-4650
*
* References to RFCs in this document identify approved and pending requests
* for change submitted to the NBS/IGES Committee. To obtain a copy of the
* documentation for an RFC, write to the following address.
*
* United States Department of Commerce
* National Bureau of Standards
* Manufacturing Systems
* Bldg. 220, Room A-353
* Washington, DC 20234

---

[1] U.S. Department of Commerce. Initial Graphics Exchange Specification (IGES), Version 2.0. Washington, D.C., 1983.

* An asterisk (*) in the left margin of this document indicates changes made
* to the specification in March, 1984.
*
* Direct your questions and comments regarding this specification to Charles
* Zonca, Chairperson of the Extensions and Repairs Subcommittee of the GM Data
* Exchange Committee, at the following address and phone number.
*
*
*

Computer Integrated Systems
Advanced Product and Manufacturing Engineering Staff
General Motors Technical Center
Engineering Staff Building
NO-Turnkey/APMES
Warren, MI 48090-9010
Phone (313) 492-1604
* (GM Network) 8-562-1604
*
* Members of the GM Data Exchange Task Force and the GM/IGES Specification
* Team contributed their time and effort to this specification document.
* Elaine Lockhart (Multiple Technologies Corporation) edited the original
* document, and Claris Henderson (also with MTC) provided production support
* for the original document.

CONTENTS

LIST OF ILLUSTRATIONS

INTRODUCTION

OVERVIEW

In 1980, the GM Engineering Computer Advisory Subcommittee of the General Technical Committee made a long-range commitment to develop a data exchange standard based on the National Bureau of Standards Initial Graphic Exchange Specification (NBS/IGES) standard. IGES is a data format that is being developed by the NBS and by representatives of industry to provide a mechanism whereby graphic systems with dissimilar data bases may exchange design data. The General Motors Integrated CAD/CAM Plan that was adopted in May, 1983, committed resources to develop a GM data exchange standard. GM's goal
* for data exchange is to ensure that 100 percent of its required CAD/CAM data
* can be exchanged among systems from GM-approved vendors.

PURPOSE OF SPECIFICATION

The purpose of this document is to provide an IGES specification that will be attached to GM purchase orders for graphic systems. The purchase orders will dictate compliance of a vendor's system to the standard. In this document, all GM-defined refinements to NBS/IGES are described in detail, and implementation dates are provided.

REQUIRED ENTITIES

The NBS/IGES Specification defines a large percentage of the GM required entities. The entities required by approved CAD/CAM systems to comply with GM standards are listed in the following tables:

- "Required NBS/IGES Geometry Entities" on page 1.2
- "Required NBS/IGES Annotation Entities" on page 1.4
- "Required NBS/IGES Structure Entities" on page 1.5

Required NBS/IGES Geometry Entities

Entity Type	Number	Form	Implementation Date
Circular Arc	100		June 1984
Composite Curve	102		June 1985
Conic Arc	104 [2]		
General Conic		0	June 1984
Ellipse		1	June 1984
Hyperbola		2	June 1984
Parabola		3	June 1984
Copious Data	106		
Points 2-D		1 [2]	June 1984
Points 3-D		2 [2]	June 1984
Linear Path 2-D		11 [2]	June 1984
Linear Path 3-D		12 [2]	June 1984
Centerline through 2-D Points		20 [2]	June 1984
Centerline through 2-D Circle		21 [2]	June 1984
* (Section 2-D, Forms 31-38, is no longer required) [3]			
Witness Line 2-D		40 [2]	June 1984
Simple Closed Area 2-D		63 [2]	June 1985
Plane	108		
Bounded Plane is Positive		1	June 1985
Bounded Plane is Negative (hole)		-1	June 1985
Line	110		June 1984
Parametric Spline Curve	112 [2]		
Linear Type 1			Dec. 1984
Quadratic Type 2			Dec. 1984
Cubic Type 3			Dec. 1984
B-spline Type 6			Dec. 1984

[2] Entity has been amended and is to be used according to the guidelines defined in this document.

* [3] This entity has been replaced by the Sectioned Area Entity (Number 230).

Required NBS/IGES Geometry Entities (cont.)

Entity Type	Number	Form	Implementation Date
Parametric Spline Surface	114 ▲		
Linear Type 1			Dec. 1984
Quadratic Type 2			Dec. 1984
Cubic Type 3			Dec. 1984
B-spline Type 6			Dec. 1984
Point	116		June 1984
Ruled Surface	118		
Equal Relative Arc Length		0	June 1985
Equal Relative Parametric Values		1	June 1985
Surface of Revolution	120		June 1985
Tabulated Cylinder	122		June 1985
Transformation Matrix	124	0	June 1984
Linear Path (see Forms 11 and 12 of Copious Data)	106 ▲		June 1984
Simple Closed Area (see Form 63 of Copious Data)	106 ▲		June 1985
Rational B-spline Curve	126 ▲		Dec. 1985
Rational B-spline Surface	128 ▲		Dec. 1985
Finite Element	136 ▪		June 1986

▲ Entity has been amended and is to be used according to the guidelines defined in this document.

▪ This entity was enhanced by the NBS/IGES Extensions and Repairs Subcommittee in February, 1984. For documentation of these enhancements, refer to RFC Number 135 for Material Properties and RFC Number 164 for Loads and Constants.

Required NBS/IGES Annotation Entities

Entity Type	Number	Form	Implementation Date
Angular Dimension	202		June 1984
Centerline (see Forms 20 and 21 of Copious Data)	106 [6]		June 1984
Diameter Dimension	206		June 1984
Flag Note	208		Dec. 1985
General Label	210		June 1984
General Note	212		June 1984
Leader (2-D)	214		
Open Triangle		1	June 1984
Triangle		2	June 1984
Filled Triangle		3	June 1984
No Arrowhead		4	June 1984
Circle		5	June 1984
Filled Circle		6	June 1984
Rectangle		7	June 1984
Filled Rectangle		8	June 1984
Slash		9	June 1984
Integral Sign		10	June 1984
Linear Dimension	216		June 1984
Ordinate Dimension	218		Dec. 1985
Point Dimension	220		Dec. 1985
Radius Dimension	222		June 1984
* Sectioned Area	230 [7]		Dec. 1985
Witness Line (see Form 40 of Copious Data)	106 [6]		June 1984

[6] Entity has been amended and is to be used according to the guidelines defined in this document.

* [7] Entity was approved by the NBS/IGES Extensions and Repairs Subcommittee
* in February, 1984. Refer to RFC Number 197 for documentation.

Required NBS/IGES Structure Entities

Entity Type	Number	Form	Implementation Date
Associativity Definition	302		
* External Reference File Index		12 [a]	Dec. 1985
Trimmed Surface		6002 [a]	Dec. 1985
Associativity Instance	402		
Group with Backpointers		1	June 1985
Views Visible		3	June 1984
Group without Backpointers		7	June 1985
Single Parent		9	Dec. 1985
* External Reference File Index		12 [a]	Dec. 1985
Trimmed Surface		6002 [a]	Dec. 1985
Drawing	404		June 1984
Line Font Definition	304		
Pointer to Subfigure		1	Dec. 1985
Repeating Structure Description		2	Dec. 1985
Property	406		
* External Reference File List		11 [a]	Dec. 1985
Offset Curve		6004 [a]	Dec. 1985
Offset Surface		6005 [a]	Dec. 1985
Trimmed Curve		6006 [a]	Dec. 1985
Subfigure Definition	308		Dec. 1984
Singular Subfigure Instance	408		Dec. 1984
View	410		June 1984
* External Reference	416 [a]		Dec. 1985

[a] Entity has been amended and is to be used according to the guidelines defined in this document.

* [a] These entities were approved by the NBS/IGES Extensions and Repairs Subcommittee in April, 1983. Refer to IGES RFC Number 136 dated September 1, 1983, for documentation of these entities.

REQUIRED REFINEMENTS

The refinements to IGES which will be required by approved CAD/CAM systems to satisfy GM's requirements are listed and described in this section. These refinements, some of which include user-defined entities and GM recommended practices, are within the guidelines and provisions of the NBS/IGES standard. GM's user-defined entities are being presented to the NBS/IGES Extensions and Repairs Committee for incorporation into the NBS standard. When these entities are incorporated, the user-defined Form numbers will be changed to the new standard NBS/IGES Form numbers. GM's required implementation dates for translator support of these entities and refinements are given in this specification. GM's vendors will be expected to support all the required entities and refinements through their IGES translators.

The following tables list the required refinements, the reason they are required, and their implementation dates. Detailed explanations of the refinements are included in subsequent sections of this document.

PARAMETERS

Refinement	Reason Required	Implementation Date
Level Number	For transferring level information between systems without restricting the number of levels.	June 1984

Required Refinements (cont.)

ENTITIES

Refinement	Reason Required	Implementation Date
Conic Arc	For transferring conic arcs in a numerically stable representation.	June 1984
Copious Data	For clarifying usage of various Forms.	June 1984
Curves and Surfaces	For handling non-uniform knot spacing.	Dec. 1984
Offset Curve	For defining curves which are offset from an existing curve.	Dec. 1985 [18]
Offset Surface	For defining surfaces which are offset from an existing surface.	Dec. 1985 [18]
Trimmed Curve	For defining trimmed curves so the pre-processor does not have to refit the curve or transfer data for an untrimmed curve.	Dec. 1985 [18]
Trimmed Surface	For defining trimmed surfaces so the pre-processor does not have to refit the surface or transfer data for an untrimmed surface.	Dec. 1985 [18]

[18] When these entities are incorporated by NBS into IGES, the user-defined Form numbers will be changed to the new standard NBS/IGES Form numbers.

NBS/IGES ENTITIES TO BE ADDRESSED IN FUTURE RELEASES

The following table lists the entities that GM will address in future releases of the GM/IGES Specification.

Entity Type	Number	Form
Copious Data	106	
Points 3-D (sextuples)		3
Linear Path 3-D (sextuples)		13
* Section 2-D		31-38
Parametric Spline Curve	112	
Wilson-Fowler Type 4		
Modified Wilson-Fowler Type 5		
Parametric Spline Surface	114	
Wilson-Fowler Type 4		
Modified Wilson-Fowler Type 5		
Transformation Matrix	124	
Cartesian Coordinates		10
Cylindrical Coordinates		11
Spherical Coordinates		12
Flash	125	
Defined by attached entity		0
Circular		1
Rectangular		2
Donut		3
Canoe		4
Node	134	

* (Finite Element Entity Number 136 is now a required geometry entity)

Associativity Instance	402	
Views Visible, Pen, Line Weight		4
Entity Label Display		5
View List		6
Signal String		8
Text Node		10
Connect Node		11

NBS/IGES Entities to be Addressed in Future Releases (cont.)

Entity Type	Number	Form
Macro Definition	306	
Macro Instance	600-699	
Property	406	
Definition Levels		1
Region Restriction		2
Level Function		3
Region Fill		4
Line Widening		5
Drilled Hole		6
Reference Indicator		7
Pin Number		8
Part Number		9
Hierarchy		10
Rectangular Subfigure Instance	412	
Circular Subfigure Instance	414	
Text Font Definition	310	

POSSIBLE FUTURE REQUIREMENTS

Gordon Surface

The Gordon Surface entity is not required at this time. It may be required if GM's graphic system cannot satisfactorily approximate Gordon surfaces with B-spline surfaces.

* Solid Modeling

```
*
*
* An RFC will be presented by the NBS/IGES Advanced Geometry Subcommittee in
* May, 1984, to extend IGES to handle solid modeling data. The solid modeling
* entities will undergo a year of testing and evaluation. During this time,
* the RFC will be open to changes to accommodate the needs of the full
* NBS/IGES community. This NBS Subcommittee will then prepare a Change Order
* to incorporate formally these entities into the NBS/IGES standard. GM is
* participating in these activities and will address the solid modeling enti-
* ties in future releases of this document.
```

ASCII FORMAT FILE STRUCTURE

Refer to the ASCII Format File Structure that is described in NBS/IGES, pp. 9-22.

Directory Entry Section

The Directory Entry (DE) Section provides an index for the file and contains attribute information for each entity. See NBS/IGES, p. 23, for further information.

Level Number

The Level Number should be used in accordance with the following GM recommended practice.

Number	Field Name	Meaning and Notes
5	Level Number	This entity is defined on this graphic display level or levels. There is no restriction on the number of levels that may be used. CAD/CAM systems must develop methods for handling a number of levels greater than their current system restrictions.

See NBS/IGES, p. 24, for further information.

CONIC ARC

This section describes a GM recommended practice that is intended to remove the numerical instability of the IGES algebraic representation of conics without changing the current representation.

Algebraic Versus Geometric Representation

The Conic Arc Entity (as defined in NBS/IGES, pp. 71-75), treats all conic arcs collectively with the algebraic representation

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0. \qquad (1)$$

For GM purposes, the recommended practice is to avoid using the algebraic representation because it has the following disadvantages when compared with the geometric representation.

- The algebraic representation is numerically unstable in certain situations. See Figure 1 on page 3.2 for an example which supports this point.

- The determination of the type of the conic by means of the algebraic representation (1) depends upon whether certain invariants are positive, zero, or negative. When you work with floating point arithmetic to evaluate the invariant, you may not get the value zero. Thus, you may risk losing the characterization of the type of the conic arc. In the geometric representation, the conic arc type is easily obtainable from the data.

- To rotate a conic in the algebraic representation, each point must be rotated independently. In the geometric representation, only the defining points and vectors are rotated.

- The graphic users, as a rule, specify geometric elements to define conic arcs, so this must be the primary data to be stored. The algebraic data, that is, the six coefficients A, B, C, D, E, F, of (1), is secondary data and can be derived when needed.

Consider the two conics (circles):

$$10^{-2} \times 0.1111(X^2 + Y^2) - 0.9999X - 0.999Y + 10^6 \times 0.4040 = 0 \qquad (2)$$

$$10^{-2} \times 0.1110(X^2 + Y^2) - 0.9999X - 0.999Y + 10^6 \times 0.4040 = 0 \qquad (3)$$

The Conics differ in the coefficients A and C only, see (1), by 10^{-6}, which is a very small quantity, but (2) is a circle with center C=(450,450) and radius R=31.62277; however, (3) is a circle with center C=(450.40541, 450.40541) and radius R=41.59964, which is a very substantial difference.

Figure 1. Example of Numerical Instability Using the Algebraic Representation: Conic Arc

Recommended Practice

To remove the numerical instability of the algebraic representation, use the geometric representations described below.

For ellipses and hyperbolas, use the geometric representation

$$Ax^2 + Cy^2 + F = 0 \qquad (4)$$

in conjunction with a matrix defining the transformation in 3-D. In this representation

$$B = D = E = 0, \quad A = b^2, \quad C = \pm a^2, \quad F = -a^2 b^2, \qquad (5)$$

where a and b are the lengths of the major and minor semi-axes. (See Figure 2 on page 3.3.)

The normalized form of (4) is

$$Ax^2 + Cy^2 + F = 0 \tag{6}$$

and $A = 1/a^2$, $C = \pm 1/b^2$, $F = -1$.

The minus sign for C is chosen when the conic is a hyperbola.

For a parabola, use the geometric representation

$$Cy^2 + Dx = 0 \tag{7}$$

in conjunction with a transformation matrix being called in by a pointer. In this representation

$$A = B = E = F = 0, \quad C = 1, \quad D = -4a,$$

and a is the distance of the vertex from the focus. (See Figure 2.)

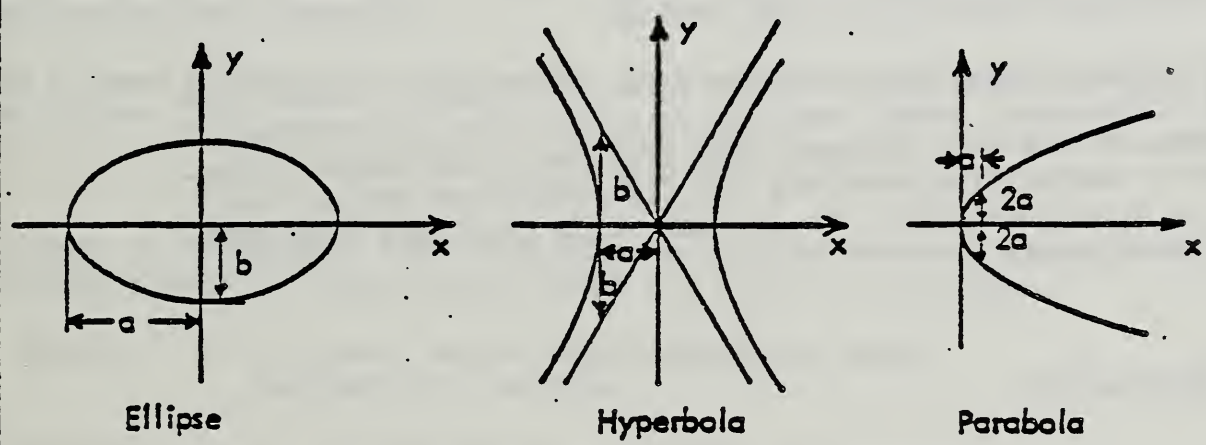


Figure 2. Graphs of Ellipse, Hyperbola, and Parabola in Standard Form

Directory Data

Entity Type Number: 104

Parameter Data

See NBS/IGES, pp. 71-75, for parameter data.

COPIOUS DATA

Clarification notes on the use of Copious Data are described here to help ensure consistent usage of various forms between different CAD/CAM systems. See NBS/IGES, pp. 76-78, for further information.

Clarification Note about Forms 1, 2, 11, and 12

Forms 1 and 2 of the Copious Data entity (106) pertain to sets of points that are not necessarily ordered. Forms 11 and 12 pertain to ordered sets of points connected consecutively by straight line segments.

Clarification Note about Form 12

GM's CGS translator maps multi-point lines into Copious Data Form 12 (3-D). This entity must be interpreted by post-processors as geometric data. The geometric operators of the vendor systems must be able to manipulate these multi-point lines geometrically. For example, the vendor systems must be able to smooth these lines with curves, generate surfaces, etc.

Clarification Note about Forms 20, 21, 40, and 63

To handle 3-D planar Centerline, Witness Line, and Simple Closed Area entities, IGES translators should map the data into the 2-D entities of Centerline (Forms 20 and 21), Witness Line (Form 40), and Simple Closed Area (Form 63) in conjunction with the 3-D Transformation Matrix (124). This practice is consistent with the way that IGES maps data from definition space to model space.

CURVES AND SURFACES

CAD/CAM systems and their IGES translators must support non-uniform knot spacing for curves and surfaces involving splines. The non-uniform knot spacing can be handled either directly or by approximating the non-uniform spacing with uniform spacing.

The following entities are affected:

- Parametric Spline Curve (Types 1, 2, 3, and 6)
- Parametric Spline Surface (Types 1, 2, 3, and 6)
- Rational B-spline Curve
- Rational B-spline Surface

This section provides information about the organization of the data. See NBS/IGES, p. 194, for further information.

ASSOCIATIVITY DEFINITION

The Associativity entities are designed for use when several entities must be logically related to one another. The Associativity Definition entity specifies the structure of the logical relationships. See NBS/IGES, p. 195, for further information.

Trimmed Surface Definition

The Trimmed Surface Definition is used to define a trimmed surface so that the pre-processor does not have to refit the surface or transfer data for an untrimmed surface.

This entity is associated with the Trimmed Surface Associativity Instance entity type 402, Form 6002.

The Trimmed Surface uses a parameter value called opacity which is defined as follows:

Opacity is a variable that is defined on surface boundary curves and/or isolated points and takes on the values -1, 0, or +1, as follows:

Opacity = +1 on closed curves constituting the outer boundary of a surface.

Opacity = -1 on closed curves constituting the boundary of a hole with non-empty interior on the surface.

Opacity = 0 on any curve or isolated point on the surface that delineates a hole of empty interior.

There are five classes of data for the Trimmed Surface Definition entity.

Class 1 has one entry for the base surface. That entry has one item, which is a pointer to the Base Surface entity.

Class 2 has one entry for the curve constituting the outer boundary of the trimmed surface: This entry has two items. One item is a pointer to the curve entity; the other is the value of the opacity on the outer boundary of the trimmed surface. The opacity value equals plus one (+1).

Class 3 has as many entries as there are disjoint closed curves delineating holes of non-empty interior on the trimmed surface. Each entry has two items. One item is a pointer to the corresponding closed curve; the other is the value of the opacity on this closed curve. The opacity value equals minus 1 (-1).

Class 4 has as many entries as curves in the interior of the trimmed surface other than those delineating holes of non-empty interior. In other words, the entries in this class are curves on the surface which delineate holes of one dimension (that is, along the curve itself) on the surface, and consequently these holes have empty interior. Each entry has two items. One item is a pointer to the corresponding curve on the surface; the other is the opacity value on the surface curve that delineates a hole with empty interior. The opacity value equals zero.

Class 5 has as many entries as isolated points in the interior of the trimmed surface. Each such point delineates a dimensionless hole on the surface. Each entry has two items. One item is a pointer to the corresponding point entity on the surface; the other item is the opacity value on the surface point that represents a dimensionless hole. The opacity value equals zero.

* The curves on the surface are to be given in terms of the parameter values of
* the parameters U and V of the surface.

Directory Data

Entity Type Number: 302
Form Number: 6002

Parameter Data

Parameter	Value	Format	Comment
1	5	Integer	Number of class definitions.
	Class 1		
2	1	Integer	Backpointers required for class 1.
3	2	Integer	Unordered class. Appearance of entries in class 1 is not significant.
4	1	Integer	Number of items in class 1. That item is a pointer to the base surface to be trimmed.
5	1	Integer	Indicates that the item consists of a pointer to a directory entry for the base surface.
	Class 2		
6	1	Integer	Backpointers required for class 2.
7	2	Integer	Unordered class.

Parameter	Value	Format	Comment
8	2	Integer	Number of items in class 2. First item is a pointer to the geometric curve constituting the outer boundary of the trimmed surface. Second item is the opacity value which is equal to 1.
9	1	Integer	Indicates that the item consists of a pointer to a directory entry for a geometric curve.
10	2	Integer	Indicates that the item consists of the opacity value.
	Class 3		
11	1	Integer	Backpointers required for class 3.
12	2	Integer	Unordered class.
13	2	Integer	Number of items per entry in class 3. First item is a pointer to the closed curve entity that delineates a hole in the trimmed surface. Second item is the value of opacity associated with this closed curve and has the value -1.
14	1	Integer	Indicates that the item consists of a pointer to a directory entry for the closed curve.
15	2	Integer	Indicates that the item consists of the opacity value.
	Class 4		
16	1	Integer	Backpointers required for class 4.
17	2	Integer	Unordered class.

Parameter	Value	Format	Comment
18	2	Integer	Number of items per entry in class 4. First item is a pointer to the curve on the surface that delineates a hole of empty interior. Second item is the value of opacity associated with the curve and has the value 0.
19	1	Integer	Indicates that the item consists of a pointer to a directory entry for the curve.
20	2	Integer	Indicates that the item consists of the opacity value.
	Class 5		
21	1	Integer	Backpointers required for class 5.
22	2	Integer	Unordered class.
23	2	Integer	Number of items per entry in class 5. First item is a pointer to the point on the surface that is a dimensionless hole on the surface. Second item is the opacity value on such a point on the surface and has the value 0.
24	1	Integer	Indicates that the item consists of a pointer to a directory entry for the point on the surface.
25	2	Integer	Indicates that the item consists of the opacity value.

ASSOCIATIVITY INSTANCE

The Associativity Instance specifies the information involved in a particular occurrence of a logical relationship between entities. See NBS/IGES, p. 197, for further information.

Trimmed Surface Instance

This entity is used to define a trimmed surface so that the pre-processor does not have to refit the surface or transfer data for an untrimmed surface.

This entity is associated with Trimmed Surface Associativity Definition entity type 302, Form 6002.

The operation producing a trimmed surface is called the trimming operation. Consider a surface entity that is a simply connected region (that is, it has an outer boundary but no inner boundary), and call it the base surface. A trimmed surface is the product of an operation on the base surface that alters the outer boundary or introduces inner boundaries to the base surface.

Altering the outer boundary of the base surface precedes introducing inner boundaries when the trimming operation involves altering the outer boundary and introducing an inner boundary to the base surface.

If the trimming of a base surface involves altering the outer boundary, then a closed curve is generated from the curve segments participating in the outer boundary of the trimmed surface. The closed curve constitutes the outer boundary of the trimmed surface.

The outer boundary of the trimmed surface might consist of only one closed curve interior to the base surface (see class a in Figure 3 on page 4.7). It might also consist of a closed curve part which is common with the outer boundary of the base surface, (see class b, c, and d in Figure 3 on page 4.7). The shaded areas in the Figure depict the trimmed surface.

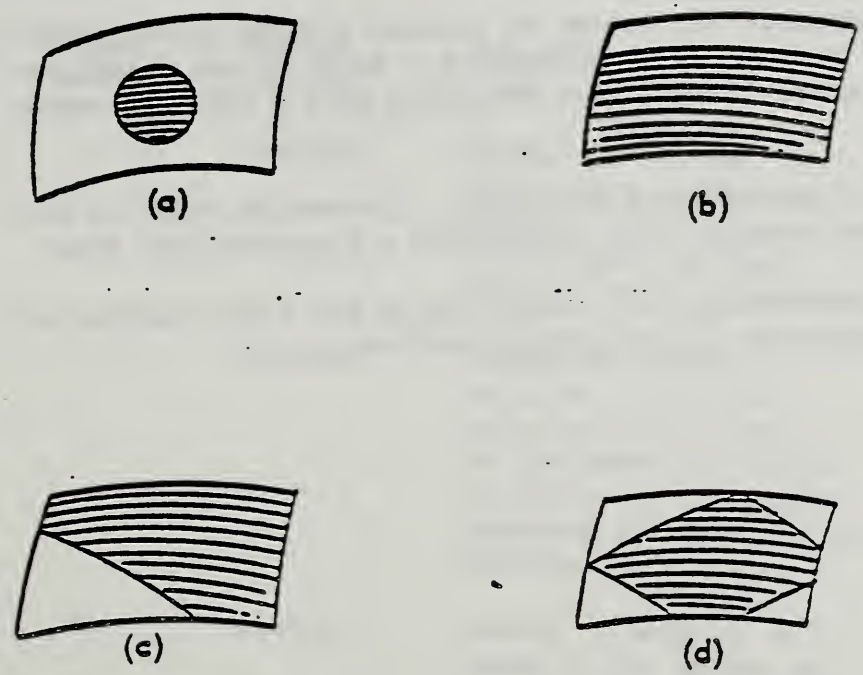


Figure 3. Classes of Trimmed Surfaces

To determine the sense of direction in tracing the curve of the outer boundary of the trimmed surface, designate one vertex of the outer boundary polygon as first, and then stipulate a sense rotational direction (clockwise or counterclockwise).

For uniformity, choose as first vertex the one with minimum u and minimum v, and choose the counterclockwise rotational direction.

If the trimming of a base surface involves introduction of an inner boundary, then you have the following cases:

- The inner boundary consists of, or has as a part, a closed curve in the interior of the trimmed surface that delineates a hole of non-empty interior.

- The inner boundary consists of, or has as a part, a curve in the interior of the trimmed surface that delineates a hole of one dimension (that is, along the curve itself) and therefore such a hole has empty interior.

- The inner boundary is, or contains as a part, a geometric point in the interior of the trimmed surface that represents a dimensionless hole.

From each of the above three categories, there may be any finite number of inner boundaries in the interior of the trimmed surface.

Directory Data

Entity Type Number: 402
Form Number: 6002

Parameter Data

Parameter	Value	Format	Comment
1	1	Integer	Number of entries in class 1. The only entry in this class is the base surface that is to be trimmed.
2	1	Integer	Number of entries in class 2. The only entry in this class is the geometric curve constituting the outer boundary of the trimmed surface.
3	N3	Integer	Number of entries in class 3. The number of disjoint closed curves delineating holes (of non-empty interior) on the trimmed surface. Default value is 0.
4	N4	Integer	Number of entries in class 4. The number of curves in the interior of the trimmed surface which delineate holes of one dimension (along the curve itself) on the surface, and consequently those holes have empty interior. Default value is 0.
5	N5	Integer	Number of entries in class 5. The number of isolated points in the interior of the trimmed surface, each representing a dimensionless hole on the surface. Default value is 0.

Parameter	Value	Format	Comment
	Class 1		
6	DES	Pointer	Pointer to the base surface that is to be trimmed.
	Class 2		
7	DEC	Pointer	Pointer to the curve constituting the outer boundary of the trimmed surface.
8	1	Integer	Opacity value assigned to the outer boundary of the trimmed surface.
	Class 3		
9	DE1	Pointer	Pointer to the first disjoint closed curve.
10	-1	Integer	First opacity value.
.	.	.	
.	.	.	
.	.	.	
7+2N3	DEN3	Pointer	Pointer to the last disjoint closed curve.
8+2N3	-1	Integer	Last opacity value.
	Class 4		
9+2N3	DE1	Pointer	Pointer to the first curve on the surface that delineates a hole of empty interior.
10+2N3	0	Integer	First opacity value.
.	.	.	
.	.	.	
.	.	.	
7+2N3+2N4	DEN4	Pointer	Pointer to the last curve.
8+2N3+2N4	0	Integer	Last opacity value.

Parameter	Value	Format	Comment
	Class 5		
9+2N3+2N4	DE1	Pointer	Pointer to the first point on the surface that is a dimensionless hole on the surface.
10+2N3+2N4	0	Integer	First opacity value.
.	.	.	
.	.	.	
.	.	.	
7+2N3 +2N4+2N5	DEN5	Pointer	Pointer to the last point on the surface.
8+2N3 +2N4+2N5	0	Integer	Last opacity value.
9+2N3 +2N4+2N5	M	Integer	Number of backpointers to associativity entities or text pointers to general notes.
10+2N3 +2N4+2N5	DE	Pointer	Pointers to associativities or general notes.
.	.	.	
.	.	.	
.	.	.	
9+2N3+2N4 +2N5+M	DE	Pointer	
10+2N3+2N4 +2N5+M	N	Integer	Number of properties.
11+2N3+2N4 +2N5+M	DE	Pointer	Pointers to properties.
.	.	.	
.	.	.	
.	.	.	
10+2N3+2N4 +2N5+M+N	DE	Pointer	

PROPERTY

Properties allow non-geometric numeric or textual information to be related to any entity. See NBS/IGES, p. 256, for further information.

Offset Curve

The offset curve property entity contains the numerical data necessary to determine the offset of a given curve. The offset curve is restricted here to apply to curves according to the following definitions.

Definition: General 3-D Curve

Let C denote a curve in the 3-D Euclidean space being analytically represented by

$$r(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix}, \quad 0 \leq t \leq 1, \tag{1}$$

where r(t) is differentiable in $0<t<1$, and regular in $0<t<1$, that is,

$$\frac{dr(t)}{dt} \text{ is not equal to 0 in the interval } 0<t<1. \tag{2}$$

Let T, N and B denote the unit vectors along the positive tangent, the principal normal (first normal), and the binormal (second normal), at a point P on the curve C (see Figure 4 on page 4.13). Also let V be a given unit vector.

Definition: Offset 3-D Curve

The offset curve entity, C_d, of a given differentiable regular curve C with parametric representation (1) is a variation of C along the lines of the field of vectors (VxT) by distance d. That is, if you denote the parametric representation of the offset curve C_d by

$$r_d(t) = \begin{bmatrix} x_d(t) \\ y_d(t) \\ z_d(t) \end{bmatrix}, \quad 0 \leq t \leq 1, \tag{3}$$

then

$$r_d(t) = r(t) + d\left(\frac{V \times T(t)}{|V \times T(t)|}\right). \tag{4}$$

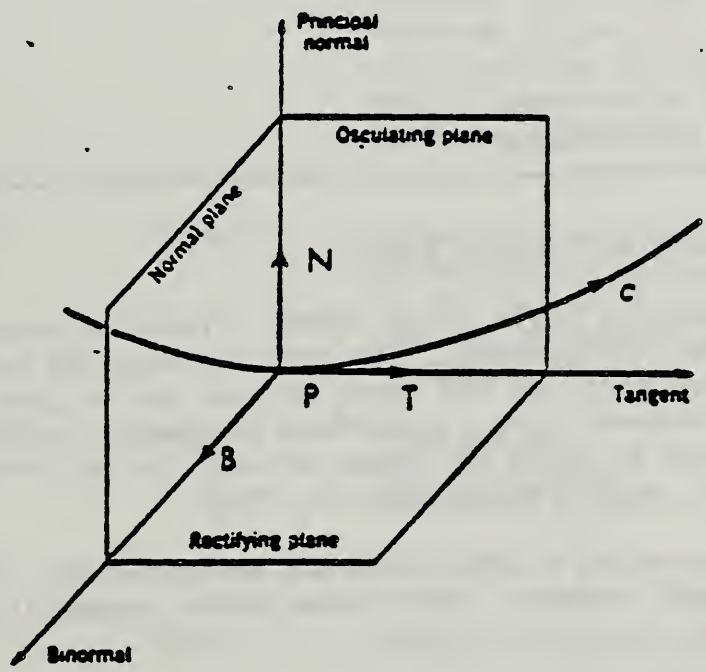


Figure 4. The Moving Trihedron Associated with a Space Curve C

Offset Planar Curves

* If the curve C is planar, then the offset curve is also planar although its
* plane, in general, is not the same as that of C, (see Figure 5a). If V is
* perpendicular to the plane of C, then VxT is a unit normal vector of C, and
* the plane of the offset curve coincides with that of C, (see Figure 5b).

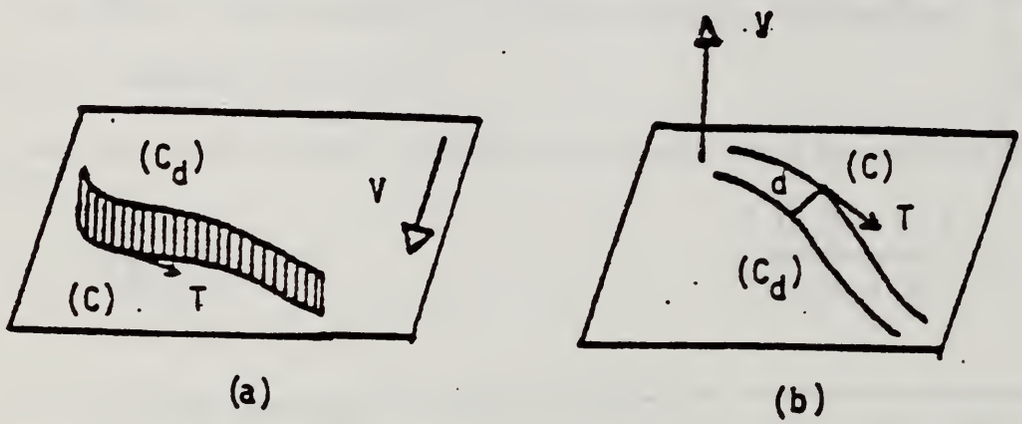


Figure 5. Offset Curve in a 2-D Plane

The IGES File for the Offset Curve Property Entity

A curve C in 3-D Euclidean space is given to be offset. This curve must be smooth and have continuous first derivatives. The data defining the curve C must allow for generating every point on the curve as well as for the tangent T at every point. It is assumed that a right-hand Cartesian coordinate system is being used. The order in which you cross multipy the unit vectors V and T is specified as V first, and T second; that is, (VxT).

The directional cosines of the vector V that determine the direction of the offset by means of the cross product (VxT) constitute numerical data entries of the Property entity.

The distance d by which the curve is offset along the lines of the field of vectors (VxT) is also a numerical data entry of the property entity.

This property is generated by an IGES pre-processor in order to process an offset curve.

Directory Data

Entity Type Number: 406
Form Number: 6004

Parameter Data

Parameter	Value	Format	Comment
1	4	Integer	Number of parameters for this property.
2	A	Floating Point	The direction cosine of V with the x-axis.
3	B	Floating Point	The direction cosine of V with the y-axis.
4	C	Floating Point	The direction cosine of V with the z-axis.
5	d	Floating Point	The distance by which the curve is offset along the lines of the field of vectors (VxT). The distance is positive in the direction of (VxT), and is negative in the opposite direction.
6	NA	Integer	Number of backpointers to associativity entities or the number of text pointers to general note entities.
7	DE	Pointer	Pointers to associativities or general notes.
.	.	.	
.	.	.	
.	.	.	
6+NA	DE	Pointer	
7+NA	MA	Integer	Number of properties.
8+NA	DE	Pointer	Pointers to properties.
.	.	.	
.	.	.	
.	.	.	
7+NA+MA	DE	Pointer	

Offset Surface

The Offset Surface Property entity contains the numerical data necessary to determine the offset of a given surface.

Mathematical Background

Let S be a subset of 3-D Euclidean space, and let S denote a surface being analytically represented by

$$r(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix}, \quad 0 \leq u,v \leq 1 \tag{1}$$

where S is regular in $0<u,v<1$, compact, and orientable. (See Figure 6.)

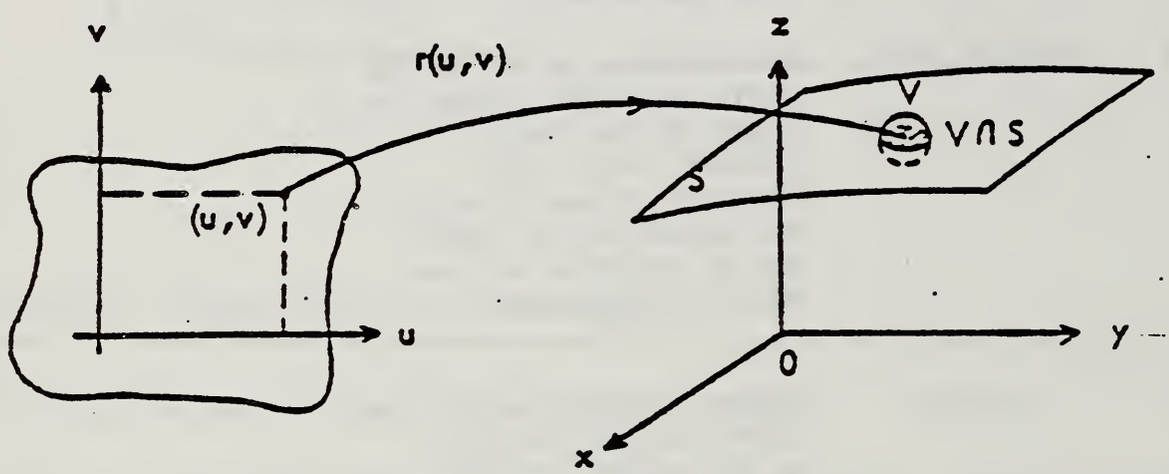


Figure 6. Parametric Representation of a Surface in 3-D Euclidean Space

The regularity of a surface is a concept whose meaning narrows or broadens according to the objective pursued. For G.M. purposes where the objective is offset surfaces, surface S must have a continuous unit normal vector at each of its points (u,v), $0<u,v<1$. The following definition of regularity handles the offset surface.

Definition of Regularity: The surface S, represented by (1), is said to be regular if

- $r(u,v)$ has continuous partial derivatives of the first order in the domain: $0<u,v<1$.

- The inverse mapping $r^{-1}(x,y,z)$ is continuous.

- For each point (u,v) in: $0<u,v<1$, the differential dr evaluated at (u,v) is one to one.

Definition of Compactness: A subset S of the 3-D geometric space is compact if it is bound and closed in the sense that it contains all its limit points. A point p is said to be a limit point of A if every 3-D neighborhood of p contains a point of A other than p. A surface S being represented by (1) and regular is compact.

Definition of Orientability: A regular surface S is orientable if, and only if, there exists a continuous field of unit normal vectors N on S.

A continuous field of unit normal vectors on an open set S means a continuous mapping N that associates to each point (u,v) of U a unit normal vector $N(u,v)$ to S at the point (u,v).

Given the representation (1) for S, you have for the unit normal vector $N(u,v)$

$$N(u,v) = \frac{r_u \times r_v}{\left| r_u \times r_v \right|}, \tag{2}$$

where r_u and r_v are the partial derivatives of $r(u,v)$

with respect to the parameters u and v.

Definition of Offset Indicator: Define as positive that orientation of S which contains the unit normal vector

$$N(0+,0+) = \frac{r_u \times r_v}{\left| r_u \times r_v \right|} \quad \text{where} \quad \lim_{u,v \to (0+,0+)} N(u,v) = N(0+,0+) \tag{3}$$

in its field of unit normal vectors to S. Call the unit normal vector N(0+,0+) the offset indicator of the surface S.

The Parametric Representation of the Offset Surface

Let S be a regular, compact, and orientable surface with parametric representation (1), which is offset normally by distance d. Let the offset surface be denoted by S_d and its parametric representation be

$$r_d = \begin{bmatrix} x_d(u,v) \\ y_d(u,v) \\ z_d(u,v) \end{bmatrix}, \quad 0 \leq u,v \leq 1. \tag{4}$$

If N(u,v) denotes the unit normal vector to S at (u,v) and you set

$$\bar{d}(u,v) = d * N(u,v), \tag{5}$$

then you have for the parametric representation of S_d

$$r_d = r + \bar{d} = \begin{bmatrix} x(u,v) + d \cos a(u,v) \\ y(u,v) + d \cos b(u,v) \\ z(u,v) + d \cos c(u,v) \end{bmatrix}, \tag{6}$$

where cos a(u,v), cos b(u,v), cos c(u,v) are the directional cosines of N(u,v).

The IGES File for the Offset Surface Property Entity

The surface S in 3-D Euclidean space is given to be offset. This surface must be regular in $0<u,v<1$, compact, and orientable.

The distance d by which the surface is offset along the unit normal vector N(0+,0+) is a numerical data entry of the property entity. It is assumed that a right-hand Cartesian coordinate system is being used.

The offset indicator of the surface is the unit normal vector N(0+,0+), at the point (0+,0+). The directional cosines of the offset indicator are numerical data entries of the property entity. (See Figure 7.)

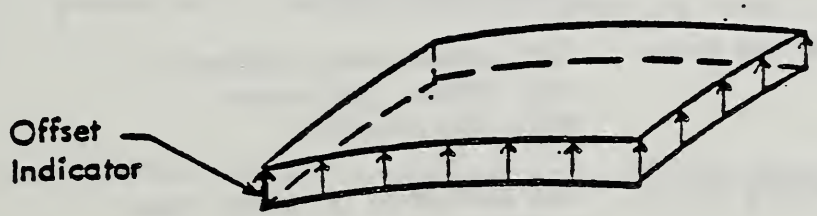


Figure 7. Offset Surface in 3-D Euclidean Space

Directory Data

Entity Type Number: 406
Form Number: 6005

Parameter Data

Parameter	Value	Format	Comment
1	4	Integer	Number of parameters for this property.
2	A	Floating Point	The directional cosine of the offset indicator with the x-axis.
3	B	Floating Point	The directional cosine of the offset indicator with the y-axis.
4	C	Floating Point	The directional cosine of the offset indicator with the z-axis.
5	d	Floating Point	The distance by which the surface is normally offset on the side of the offset indicator if d>0; andon the opposite side if d<0.
6	NA	Integer	Number of backpointers to associativity entities or the number of text pointers to general note entities.
7	DE	Pointer	Pointers to associativities or general notes.
.	.	.	
.	.	.	
.	.	.	
6+NA	DE	Pointer	
7+NA	MA	Integer	Number of properties.
8+NA	DE	Pointer	Pointers to properties.
.	.	.	
.	.	.	
.	.	.	
7+NA+MA	DE	Pointer	

Trimmed Curve

Given a curve in some analytical form with its boundary points, the need may arise to trim it on either end by a certain amount. If the curve is given analytically either by one polynomial expression throughout its domain or by means of a set of points to be interpolated by some underlying interpolation scheme then trimming would not require the introduction of a separate entity. In this case, simply redefine the end-points of the curve to produce the desired curve.

However, if the curve is given analytically by a spline function, trimming a curve and discarding the trimmed-off end parts require refitting the spline because the trimming may not occur on the knots.

To avoid refitting a spline curve, transfer the existing spline along with the two points on it where the trimming is to occur. Those points are given by their corresponding parameter values which constitute the numerical data contained in this property.

This property is generated by an IGES pre-processor in order to process a trimmed curve.

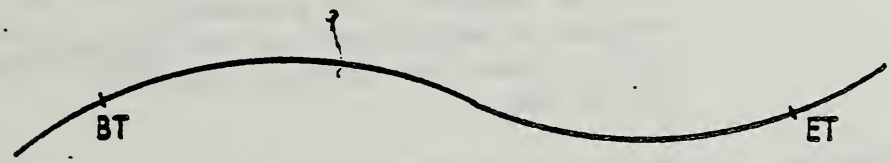


Figure 8. Trimmed Curve

Directory Data

Entity Type Number: 406
Form Number: 6006

Parameter Data

Parameter	Value	Format	Comment
1	2	Integer	Number of parameters for this property.
2	BT	Floating Point	The value of the parameter of the curve to be trimmed at the beginning point of the trimmed curve (beginning in the direction of increasing parameter values). The default value would be the beginning parameter value of the base curve. If the range of the parameter values is normalized in the interval (0,1), then the default value of BT is zero.
3	ET	Floating Point	The value of the parameter of the curve to be trimmed at the ending point of the trimmed curve. The default value would be the ending parameter value of the base curve. If the range of the parameter values is normalized in the interval (0,1), then the default value of ET is one.
4	NA	Integer	Number of backpointers to associativity entities or the number of text pointers to general notes.
5	DE	Pointer	Pointers to associativities or general notes.
.	.	.	
.	.	.	
4+NA	DE	Pointer	
5+NA	MA	Integer	Number of properties.
6+NA	DE	Pointer	Pointers to properties.
.	.	.	
.	.	.	
5+NA+MA	DE	Pointer	

APPENDIX A. MEMBERS OF THE GM DATA EXCHANGE COMMITTEE

Atallah, Paul
Cadillac Motor Car Div., Dept 4017
Computer Systems
2860 Clark Avenue
Detroit, MI 48232-COURIER

8-284-6601

Bachelor, Paul
Hydra-Matic Division
Ecorse & Wiard Roads
Ypsilanti, MI 48197
COURIER

8-335-6695

Bowen, Sandra
N2 - CIS
APMES Engineering North
GM Tech Center
Warren, MI 48090 - 9010

8-535-1181

Cenowa, Ron
Fisher Body
Computer Systems 192-32
GM Tech Center
Warren, Mi 48090 - 9010

(313) 575-8936

Georges, Stacy
Central Foundry Div.
77 West Center St.
Saginaw, MI 48605-COURIER

8-386-3495

Loewengruber Kontry, Karen
No-Turnkey
APMES - Engineering North
GM Tech Center
Warren, MI 48090 - 9010

8-562-1604

Meise, Konrad
APMES - MD 69
Adam Opel Contact
GM Tech Center
Warren, MI 48090 - 9010

(313) 492-0433

* Murray, Lyle 8-387-5572
* AC Spark Plug
* 1300 N. Dort Hwy.
* Flint, MI 48556-COURIER
*
*
* Nelson, Ron 8-356-3541
* GMC-Inland Division
* 2701 Home Avenue
* Dayton, OH 45401-COURIER
*
*
* Norfleet, Larry (313) 492-1688
* APMES - CIS
* 6454 E. 12 Mile Road
* GM Tech Center
* Warren, MI 48090 - 9010
*
*
* Panzica, Connie (313) 492-1935
* GMAD
* 30009 Van Dyke
* GM Tech Center
* Warren, MI 48090 - 9010
*
*
* Rank, Charles 8-386-5822
* Saginaw Steering Gear
* Prod. Eng.-Plant #3
* 3900 Holland Rd.
* Saginaw, MI 48605-COURIER
*
*
* Senft, Erich (313) 456-2368
* Truck & Bus
* 660 South Blvd. E.
* Pontiac, MI 48053-COURIER
*
*
* Spewock, Nicholas (313) 575-1181
* Engrg Bldg.
* N2-CIS
* GM Tech Center
* Warren, MI 48090 - 9010

* Vlaeminck, Leon (313) 857-1683
* Pontiac Motor Division
* Information Systems
* Pontiac, MI 48053
*
*
* Williams, Dan (313) 575-3640
* Chevrolet Eng., A-244
* GM Tech Center
* Warren, MI 48090 - 9010
*
*
* Zonca, Charles (313) 492-1604
* No-Turnkey
* APMES Engineering North
* GM Tech Center
* Warren, MI 48090 - 9010

APPENDIX B. MEMBERS OF THE GM EXTENSIONS AND REPAIRS SUB-COMMITTEE

Bowen, Sandra 8-535-1181
N2 - CIS
APMES-Engineering North
GM Tech Center
Warren, MI 48090 - 9010

Dick, Arnold (313) 575-3595 or 8-535-3595
Chevrolet Motor Division
CEC- Rm. A-239
GM Tech Center
Warren, Mi 48090 - 9010

Grabowski, Joseph (313) 554-6604
Cadillac Motor Car Div., Dept 4017
2860 Clark Avenue
Detroit, MI 48232 - COURIER

Greenewald, Paul 8-387-8334
AC Spark Plug
1300 North Dort Hwy
Flint, MI 48556 - COURIER

Hinton, William (313) 456-4744
World Truck & Bus Div.
Eng. Operations
660 South Blvd. E.
Pontiac, MI 48053-COURIER

Loewengruber Kontry, Karen (313) 492-1604
NO-Turnkey
APMES-Engineering North
GM Tech Center
Warren, MI 48090 - 9010

Meise, Konrad (313) 492-0433
APMES - MD 69
Adam Opel Contact
GM Tech Center
Warren, MI 48090 - 9010

* Nicholas, Judy (313) 575-0698
* APMES - CIS
* Environmental Activities Bldg.
* GM Tech Center
* Warren, MI 48090 - 9010
*
*
* Trafidlo, Heidi 8-237-1500
* Pontiac Motor Div.
* Pontiac, MI - COURIER
*
*
* Tsimis, Emmanuel (313) 575-4082 or 8-535-4082
* Fisher Body
* Engineering 192-32
* GM Tech Center
* Warren, Mi 48090 - 9010
*
*
* Wethington, Robert (313) 575-1181
* N2 - CIS
* Engineering North
* GM Tech Center
* Warren, Mi 48090 - 9010
*
*
* Witwer, Mel 8-358-7707
* Delco Products
* Mail Stop 4-07
* P.O. Box 1042
* Dayton, OH 45401-COURIER
*
*
* Woelfel, Tom (313) 575-2247
* APMES - CIS
* Environmental Activities Bldg.
* GM Tech Center
* Warren, MI 48090 - 9010
*
*
* Zonca, Charles (313) 492-1604 or 8-562-1604
* NO-Turnkey
* APMES - Engineering North
* GM Tech Center
* Warren, MI 48090 - 9010

Inter-Organization

Advanced Product and
Manufacturing Engineering Staff

General Motors Corporation
General Motors Technical Center
Warren, Michigan 48090-9040

Date: March 30, 1984

To: Holders of GM/IGES Specification

From: Ruth Y. Reed

Subject: Revisions

Enclosed are revisions for the GM/IGES Specification (CGS 101-F-01) that was first published in December, 1983.

Update your manual by adding (A) and replacing (R) these pages:

R	Title Page
R	v - viii
R	1.1 - 1.10
R	4.1 - 4.2
R	4.13 - 4.14
R	5.1 - 5.2
A	5.3 - 5.4
R	6.1 - 6.2

An asterisk (*) in the left margin and the date March 30, 1984, denote revised pages.

Please file this memo at the back of the manual for future reference.

Ruth Y. Reed

Ruth Y. Reed, Coordinator
Technical Documentation Group
Computer Integrated Systems
FB 192-32, Ext. 5-8729

RYR/eel

Enclosures

APPENDIX C
****

Hughes/EDSG
IGES Requirements
Specification
Version 0.5

preliminary

April 1, 1986

Table of Contents

Table of Contents - Continued **Page**

plotted. Line weights have distinct meanings in design/drafting applications.

R3.3.5 Processors shall support the Line Weight Number for all entities.

3.3.6 Color Number. This value maintains eight color values for displayed entities. The color definition entity may be used to more accurately represent the colors required. However, at this time the color differentiation is more important than the visual accuracy of that color.

R3.3.6 Processors shall support the Color Number for all entities.

3.3.7 Parameter Record Count. Data integrity in the IGES file must be maintained. This field indicates the number of Parameter Data records that are used to represent the entity. This record count must accurately represent the parameter record counts to assure valid post-processing of the data.

R3.3.7 Processors shall support the Parameter Record Count for all entities.

3.3.8 Entity Label and Subscript. Some systems utilize a tag and subscript mechanism to reference specific entities in the native system. Where these mechanisms exist, the tag and/or subscript value shall be passed into the appropriate Directory Entry field.

R3.3.8-1 Processors shall support the Entity Label field as it applies to the processing system.

R3.3.8-2 Processors shall support the Entity Subscript field as it applies to the processing system.

3.4 Application Entity Sets

It is a requirement that all CAD/CAM systems used at Hughes/EDSG have the capability to exchange design data with other systems in the design cycle. To ensure that entitiy requirements are representative of the actual work flow at Hughes/EDSG, subsets of the IGES entities have been divided into Application Entity Sets. These Entity Sets are composed of entities which are required for each particular design function and which will be transferred to other design functions.

R3.4-1 CAE/CIM Systems shall support the Application Entity Set(s) defined below for each area in which the system is intended to be used.

The application entity sets addressed are: Solid Modeling, Finite Element Modeling, Mechanical Design/Drafting, Printed Circuit Board Design/Drafting, Manufacturing Process Planning, and NC Tool Path Generation. The IGES entities which are required for these entity sets are detailed below. Each application has a set of entities which processors must be able to read and write. In some cases, a select set of entities are designated as read-only. This insures that data may be captured by the application which is not required to be transfer to another application or system. In addition to the entities required by the application sets

listed later in this section, the following requirement shall apply:

R3.4-2 Any entity written out to IGES format must be able to be read by the same system's IGES Read processor.

This assures that data generated on a system can be read back into that system for data verification purposes.

3.4.1 Solid Modeling. The Solid Modeling application is used for the conceptual design of mechanical devices as well as some printed circuit board layout applications. The solid model allows the user to check for clearance problems, generate shaded images and of the product. The data generated will be used as input to the Finite Element Modeling application and other analysis packages. When the design has been captured and analyzed, the model definition will be passed to the design/drafting application and the NC/Tooling application. The solid modeling system must then be capable of generating two types of IGES output files - View dependent "picture files" for use by the design/drafting application, and surfaced 3-D wireframe representations for the NC/Tooling application.

The data generated for the design/drafting application will contain wire-frame geometry with view dependent characteristics. This will allow the creation of hidden line views within the solid modeler that can be transferred to the design/drafting systems. The data generated for the NC/Tooling application however, will contain a fully surfaced three dimensional wire frame model.

3.4.1.1 Solid Modeling Entity Set.

R3.4.1.1 Solid Modeling systems shall support all entities defined

Processing Requirement: Read/Write

Type	Form	Description
100	0	Circular arc
102	0	Composite curve
104	0	Conic arc - general form
104	1	Conic arc - ellipse
104	2	Conic arc - hyperbola
104	3	Conic arc - parabola
108	-1	Planar hole - Bounded
108	0	Plane
108	1	Bounded Plane
110	0	Line
112	0	Parametric spline curve
114	0	Parametric spline surface
116	0	Point
118	0	Ruled surface (Arc length parametrization)
118	1	Ruled surface (Equiparametric parametrization)

120	0	Surface of Revolution
122	0	Tabulated cylinder
124	0	Transformation matrix
402	3	Views visible instance
402	4	Views visible, Color, Line weight instance
402	6	View list instance
402	7	Group without back-pointers instance
402	9	Single parent instance
410	0	View - Orthographic Parallel

Processing Requirement: Read Only

Type	Form	Description
402	1	Group instance with back-pointers

3.4.2 Finite Element Modeling.

3.4.2.1 Finite Element Modeling Entity Set.

Processing Requirement: Read/Write

Type	Form	Description

To be addressed at a later date

3.4.3 Mechanical Design/Drafting. The Mechanical Design/Drafting system must be able to produce MIL Spec drawings in IGES format. This requirement includes the support of toleranced dimensions, views, drawings, and the general symbol entity for support of feature control symbols. The data produced for such a drawing must be able to be read back into the pre-processing system for editing. For a 3-dimensional system, the data will be composed of a single representation of the model and annotated views of the model which are referenced by the drawing entity.

3.4.3.1 Mechanical Design/Drafting Entity Set.

Processing Requirement: Read/Write

Type	Form	Description
100	0	Circular arc
102	0	Composite curve
106	20	Centerline
106	21	Centerline through circle centers
106	31	Section lines - General use, iron, brick, stone masonry
106	40	Witness line
110	0	Line
112	0	Parametric spline curve
116	0	Point
124	0	Transformation matrix
202	0	Angular dimension
206	0	Diameter dimension
210	0	General label
212	0	General note - Font Code 1
212	0	General note - Font Code 1001
212	0	General note - Font Code 1002
214	x	Leader arrow - 2 pts.
214	x	Leader arrow - 3 pts.
214	x	Leader arrow - 4 pts.
214	1	Leader arrow - Wedge
214	2	Leader arrow - Triangle
214	4	Leader arrow - No arrowhead

Support of Cross-hatching as per IGES Version 3.0.

.4.3.1 Mechanical Design/Drafting Entity Set (continued).

Processing Requirement: Read/Write (continued)

Type	Form	Description
216	0	Linear dimension
218	0	Ordinate dimension
222	0	Radius dimension
228	0	General symbol
308	0	Subfigure Definition
402	7	Group without back-pointers instance
404	0	Drawing
408	0	Single subfigure instance
410	0	View - Orthographic parallel

Processing Requirement: Read Only

Type	Form	Description
104	0	Conic arc - general form
104	1	Conic arc - ellipse
104	2	Conic arc - hyperbola
104	3	Conic arc - parabola
106	1	Copious data - coordinate pairs
106	2	Copious data - coordinate triples
106	11	Copious data - Piecewise planar, linear string(2D linear path)
106	12	Copious data - Piecewise linear string(3D linear path)
108	-1	Planar hole - Bounded
108	0	Plane
108	1	Bounded Plane
212	0	General note - Font Code 0
402	1	Group instance with back-pointers
402	9	Single parent instance

3.4.4 Printed Circuit Board Design/Drafting. The Printed Circuit Board Design/Drafting system must be able to support schematic capture, the physical description of the board, and the annotated drawing of the board. The system must be able to read in any data it creates for the purpose of editing. Although it is not a requirement at this time, future extensions of this specification will require the use of external symbols for the support of component libraries.

3.4.4.1 Printed Circuit Board Design/Drafting Entity Set.

Processing Requirement: Read/Write

Type	Form	Description
100	0	Circular arc
102	0	Composite curve
106	11	Copious data - Piecewise planar, linear string(2D linear path)
106	12	Copious data - Piecewise linear string(3D linear path)
106	13	Copious data - Piecewise linear string (sextuples)
106	63	Simple closed area
108	-1	Planar hole - Bounded
108	0	Plane
108	1	Bounded Plane
110	0	Line
116	0	Point
124	0	Transformation matrix
125	0	Flash - defined by attached entity
125	1	Flash - circular
125	2	Flash - rectangular
125	3	Flash - donut
125	4	Flash - canoe
132	0	Connect point
202	0	Angular dimension
206	0	Diameter dimension
210	0	General label
212	0	General note - Font Code 1
212	0	General note - Font Code 1001
212	0	General note - Font Code 1002
214	x	Leader arrow - 2 pts.
214	x	Leader arrow - 3 pts.
214	x	Leader arrow - 4 pts.
214	1	Leader arrow - Wedge
214	2	Leader arrow - Triangle
214	4	Leader arrow - No arrowhead
216	0	Linear dimension
218	0	Ordinate dimension
222	0	Radius dimension

3.4.4.1 Printed Circuit Board Design/Drafting Entity Set (continued).

Processing Requirement: Read/Write (continued)

Type	Form	Description
228	0	General symbol
302	5xxx	Associativity Definition
308	0	Subfigure definition
312	0	Text template - absolute coordinates
312	1	Text template - relative coordinates
320	0	Network subfigure definition
402	1	Group instance with back-pointers
402	7	Group without back-pointers instance
402	8	Signal string instance
402	10	Text node instance
402	11	Connect node instance
402	18	Flow instance
404	0	Drawing
406	2	Property - Region restriction
406	6	Property - Drilled hole
406	7	Property - Reference designator
406	8	Property - Pin number
406	9	Property - Part number
406	11	Property - Tabular data
406	12	Property - External reference file list
406	15	Property - Name
408	0	Single subfigure instance
410	0	View - Orthographic parallel
412	0	Rectangular subfigure instance
414	0	Circular subfigure instance
420	0	Network subfigure instance

Processing Requirement: Read Only

Type	Form	Description
104	0	Conic arc - general form
104	1	Conic arc - ellipse
104	2	Conic arc - hyperbola
104	3	Conic arc - parabola
106	1	Copious data - coordinate pairs
106	2	Copious data - coordinate triples
112	0	Parametric spline curve
212	0	General note - Font Code 0
402	9	Single parent instance

3.4.5 Manufacturing Process Planning.

3.4.5.1 Manufacturing Process Planning Entity Set.

Processing Requirement: Read/Write

Type	Form	Description
To be addressed at a later date		

3.4.6 NC Tool Path Generation.

3.4.6.1 NC Tool Path Generation Entity Set.

Processing Requirement: Read/Write

Type	Form	Description
To be addressed at a later date		

APPENDIX D
****

DRAFT PROPOSAL

TOOL DEVELOPMENT FOR IGES TRANSLATOR VERIFICATION

Computer-Aided Design Branch
Automation Systems Laboratory
Corporate Research & Development
General Electric Company

October 2, 1986

DRAFT PROPOSAL

TOOL DEVELOPMENT FOR IGES TRANSLATOR VERIFICATION

1. INTRODUCTION

The IGES translator verification is important to both CAD/CAM users and vendors. It is getting strong attention by the IGES community and the National Bureau of Standards (NBS). The IGES Testing Methodology Committee has prepared a working paper on Documentation for the Testing Methodology of IGES Translators and IGES formatted Data Files.

The General Electric Corporate Research and Development (CRD) has been analyzing the whole process of IGES translator verification to identify technical voids which need to be filled to achieve an effective verification. This draft proposal is a result of this analysis.

The target of this proposal is to develop software tools required for the IGES translator verification process in order to increase its degree of accuracy, consistency, and automation.

The next section provides a definition and description of the verification process steps including an identification of needed tools. Section 3 includes description of these tools. Section 4 details the proposed statement of work including deliverables, and schedules.

2. FUNCTIONAL DESCRIPTION OF THE VERIFICATION PROCESS

The IDEF0 methodology is used to model the functions covered by the IGES translation verification process. Figures 1 through 4 are IDEF0 charts for different levels of that process.

Figure 1 represents the top level of the IGES translator verification process. The process consists of the following three functions:

2.1 Upgrade IGES Test Library to Current Version

The NBS possesses the beginning of an extensive library of IGES test cases. These consist of a partial entity library originally donated by Boeing, the Air Force PDDI test tapes, the AUTOFACT demonstration tapes, and miscellaneous tapes donated by members of the IGES community. None of these IGES files exist in a format identical to IGES Version 3.0. Many contain syntax errors. It is necessary to correct these errors and upgrade the files to IGES Version 3.0. It will also be necessary in the future to be able to update that library to the then-current version of IGES.

There is a need for a software tool which can be used to upgrade the library to the current version. That tool is referred to as Tool #1 in Figure 1.

2.2 Test Vendor's IGES Translators

This function includes all the steps for testing vendor's IGES preprocessor and postprocessor. The inputs to this function are the current IGES version library files, and the outputs are the test results for the vendor's processors. Four different tools have been identified as required for this function and will be described below.

Figure 2 shows the function of testing vendor's IGES translators. This function consists of the following three sub-functions:

2.2.1 Generate Reference IGES File

This function generates the IGES file needed for testing the vendor's translators. There is a need for a tool (Tool #2) which can generate reference IGES files from the current version library files based on the IGES subset to be tested. The tool is expected to generate the file either by adding entity files from a library or by reducing the number of entities in a given file. An information model for the IGES subset is assumed to be available and will be used in controlling the contents of the reference file.

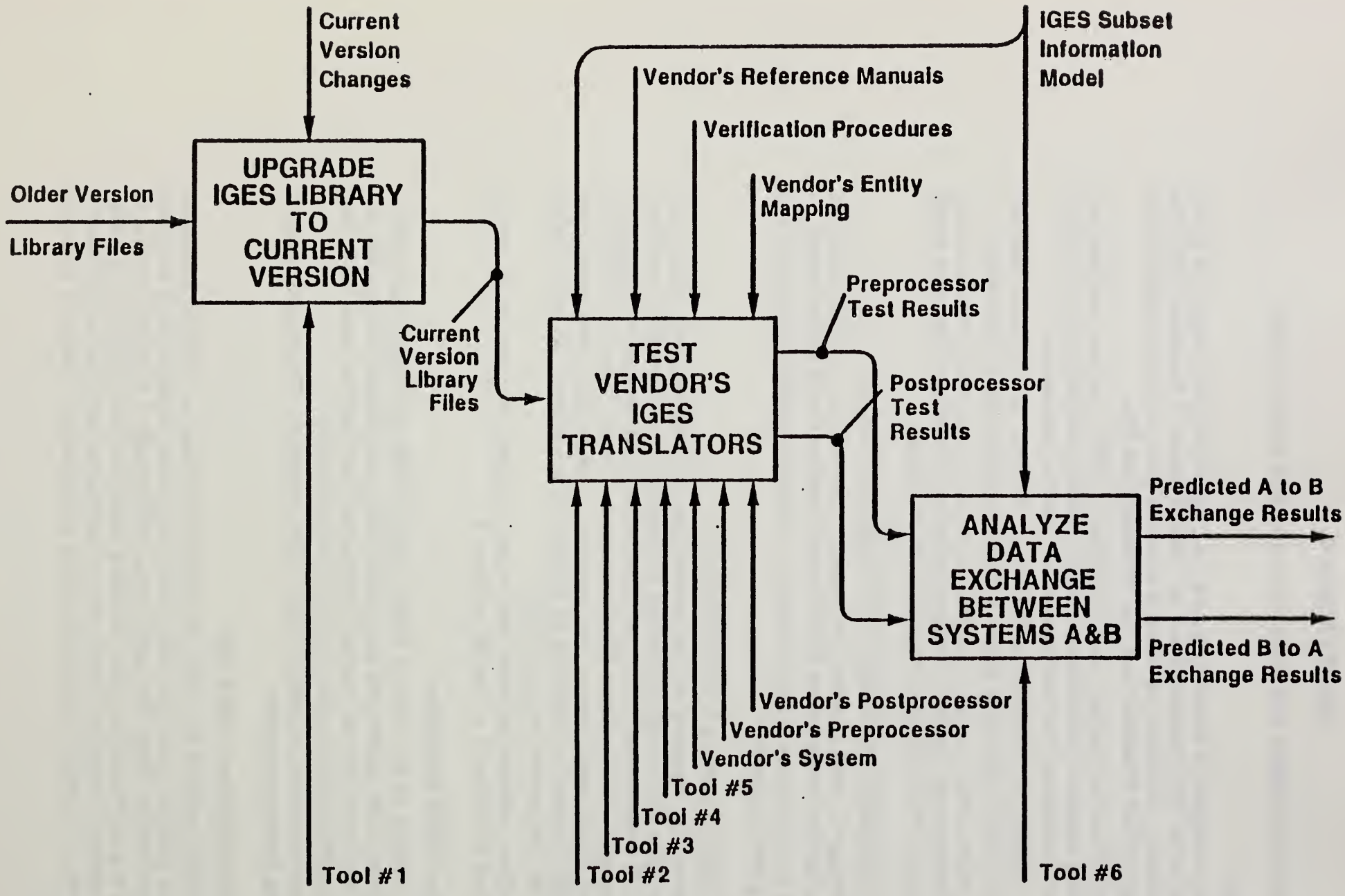


Figure 1 - VERIFY IGES TRANSLATORS

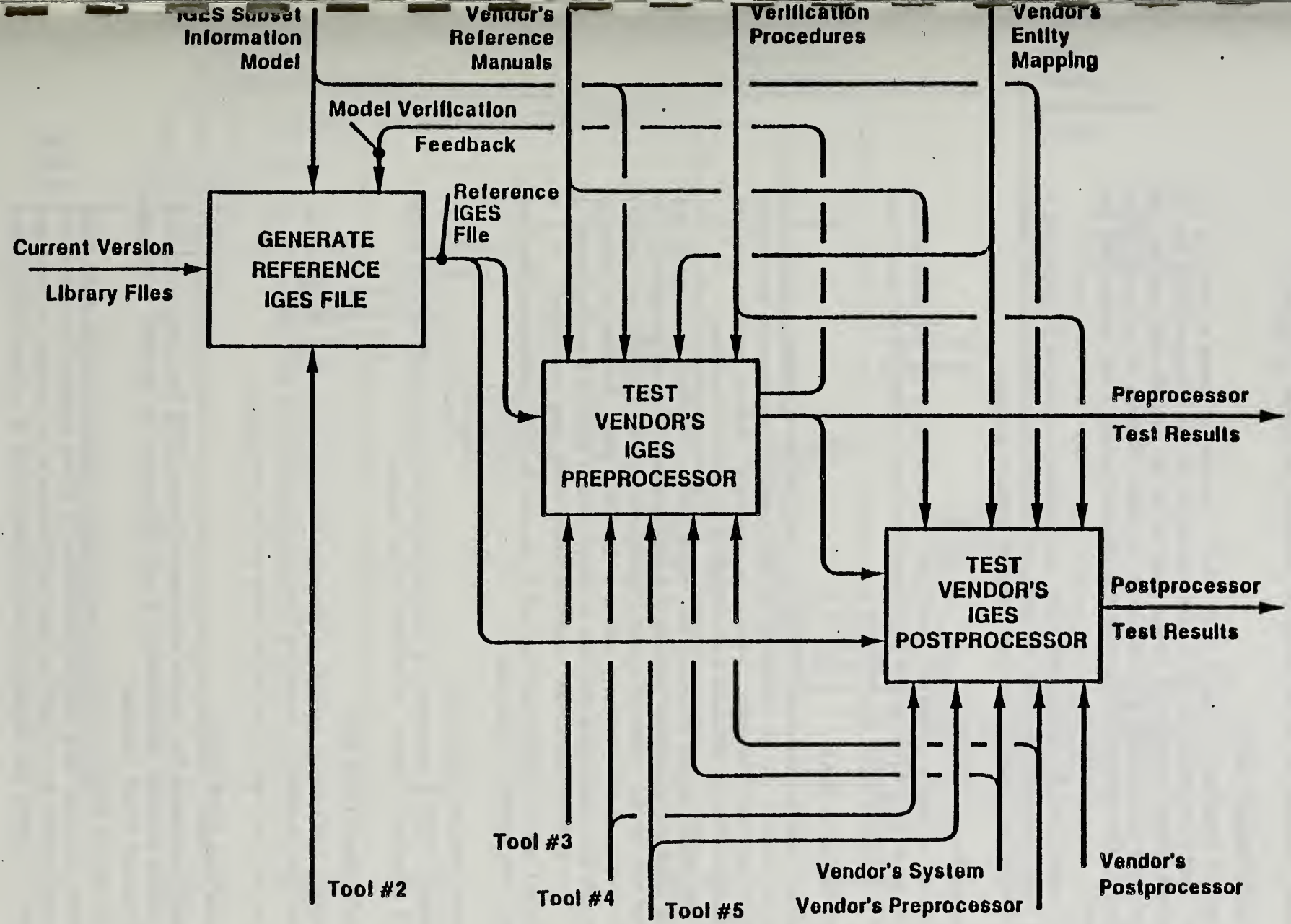


Figure 2 - TEST VENDOR'S IGES TRANSLATORS

2.2.2 Test Vendor's IGES Preprocessor

This function handles the testing of the preprocessor and generates the preprocessor test results. It consists of the following six functions: (as shown in Figure 3)

a) Generate Vendor's Script

This function generates the input script which is needed to create a model in the vendor's system equivalent to the model represented by the reference IGES file. It is extremely difficult to manually write a script which will be sufficiently precise as to allow the same functionality to be placed in the native database of any system to be encountered. One must at least expect that different operators will interpret the file differently.

There is a need for a tool (Tool #3) which can translate a reference IGES file into a specific vendor's script. Vendor's reference manuals and the IGES subset information model will be needed to provide controlling input to that tool.

b) Generate Vendor's Model

This function creates a model in the vendor's system based on the input script. There are no new tools needed for the execution of this function. The vendor's system and reference manuals are required.

c) Verify Vendor's Model

It is desirable that the model generated in the previous step is verified to ensure that the entities claimed to be supported by the vendor's preprocessor are created correctly in the native database. It would be a waste of effort to continue with the test if there are obvious problems with entities in the vendor's database.

If the verification reveals reference file related errors, feedback is passed to the reference IGES file generation function to create another file which consists of entities supported by the vendor's system. Another script is generated and another model is created. If the errors are related to the input script, feedback is passed to the vendor's script generating function to create another script and another model is created. This loop continues until a positive model verification takes place.

The vendor's system utilities are expected to be used for the verification. There are no special tools needed for this task.

d) Generate Vendor's IGES File

The vendor's IGES preprocessor is used to create an IGES file from the model created in function (b) above. No other tools are needed.

e) Verify Vendor's IGES File

The vendor's IGES file is verified to detect any syntax errors. The results of the verification are generated in a standard format and passed to the next step.

There is a need for a tool (Tool #4) to carry out that verification.

f) Compare Two IGES Models

This function compares two models represented by two IGES files. The two models should include identical information and any differences are included in the test results generated by this function.

IGES files may be physically different but acceptable within the scope of the IGES specification. That may be a result of differences in native databases or in interpreting the specification. An example of such a difference is the practice of some programmers in aligning the parameter data section values in columns while other

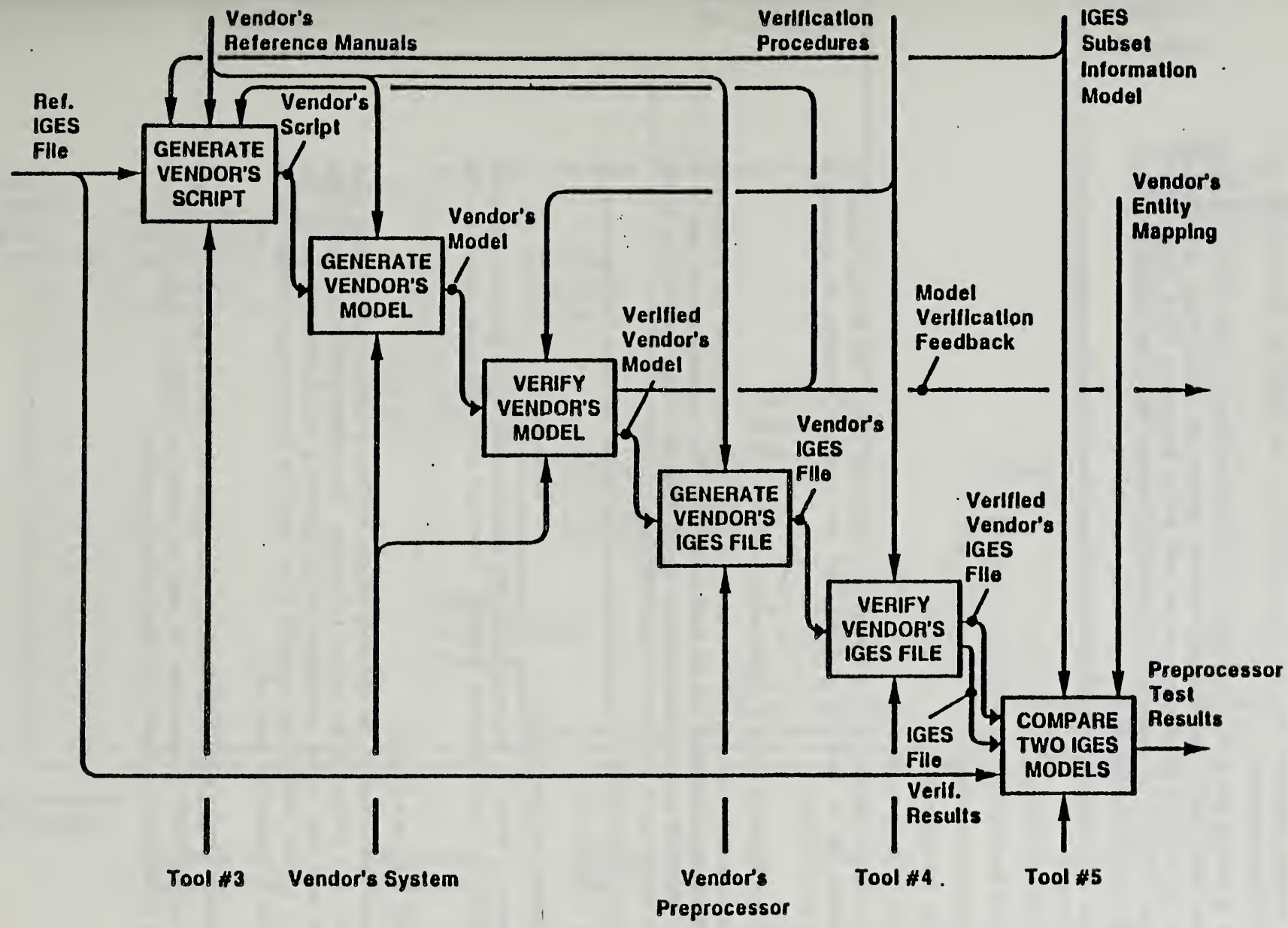


Figure 3 - TEST VENDOR'S IGES PREPROCESSOR

programmers produce as compact a representation as possible. Another example is the different ways in which entities can be ordered within an IGES file. Such differences would make a text file comparison between the files worthless, while they may represent the same information.

This function compares the reference IGES file to the IGES file generated by the vendor's preprocessor.

There is a need for a tool (Tool #5) to load the contents of each file in a neutral database then verify the database contents against the IGES subset information model to arrive at what happened to each entity during the IGES preprocessing function. Vendors are expected to provide information about the mapping of their entities into IGES entities.

The test results classify the translation outcome for each entity; examples of these classes are:

- Full functionality, entity same
- Full functionality, entity/entities different
- Partial functionality
- No functionality
- Entity not processed

Test results are generated in a report form and a file form. The file containing the test results will be used for analyzing the data exchange between two systems as shown in 2.3 below.

2.2.3 Test Vendor's IGES Postprocessor

The postprocessor test will have to be done after the preprocessor test because the the preprocessor, translator will be used in the test. The preprocessor test results are needed to figure out any postprocessor errors. The use of the preprocessor in this test avoids having to access the vendor's database directly.

The function of testing the postprocessor consists of the following five functions: (as shown in Figure 4)

a) Generate Vendor's Model

The vendor's model is generated using the postprocessor which translates the reference IGES file into the native database. There are no special tools needed for this task.

b) Verify Vendor's Model

The model generated in the previous step is verified to identify any obvious problems with the entities in the vendor's database. The results of the model verification may be used later in the comparison of IGES models described in (e) below. The vendor's system utilities are expected to be used for the verification. There are no special tools needed for this task.

c) Generate Vendor's IGES File

The vendor's IGES preprocessor is used to create an IGES file from the model created in function (b) above. No other tools are needed.

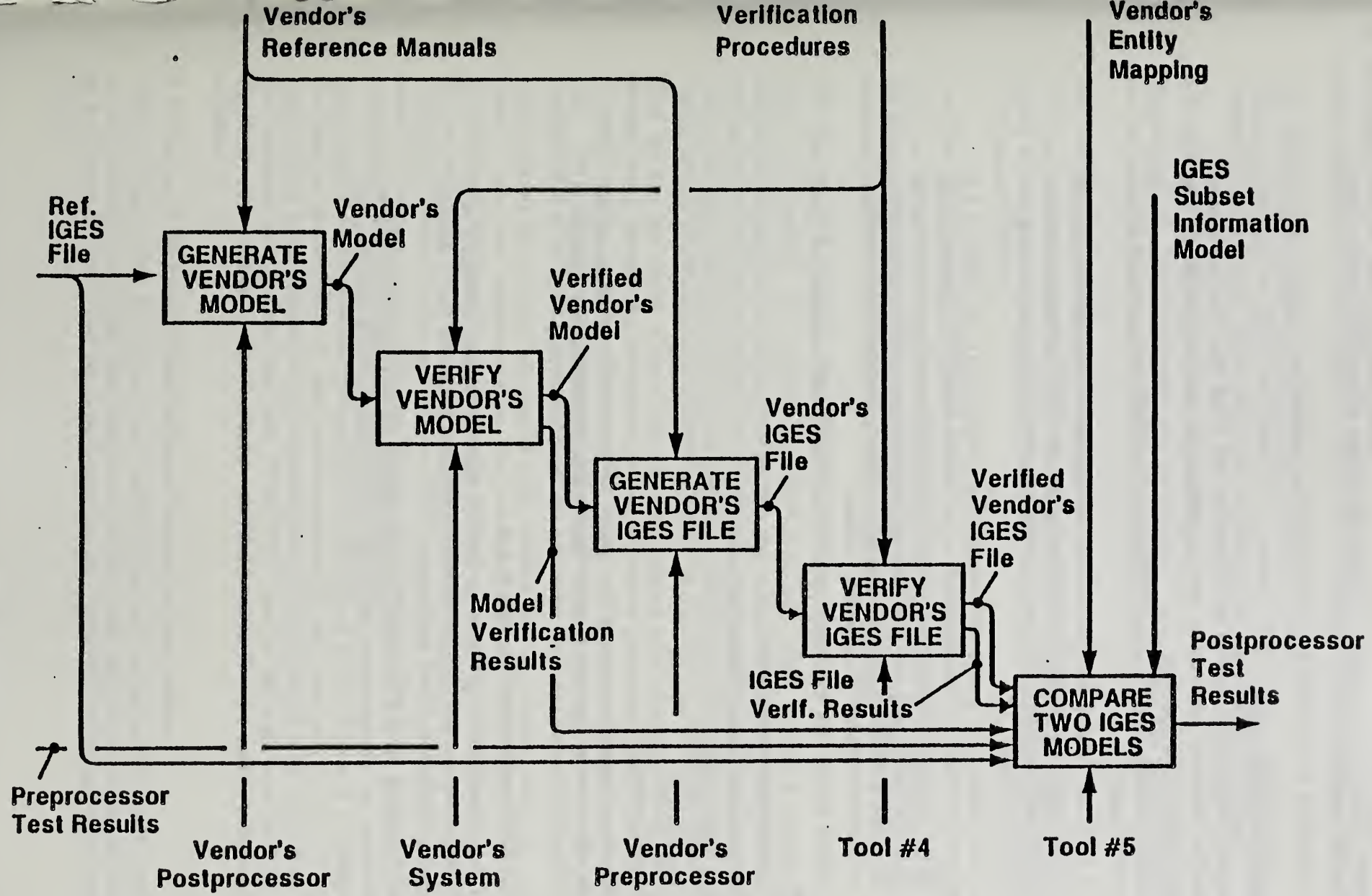


Figure 4 - TEST VENDOR'S IGES POSTPROCESSOR

d) Verify Vendor's IGES File

The vendor's IGES file is verified to detect any syntax errors. The results of the verification are passed to the next step (e).

There is a need for a tool (Tool #4) to carry out that verification.

e) Compare Two IGES Models

This function compares two models represented by two IGES files. The two models should include identical information and any differences are included in the test results generated by this function.

This function compares the reference IGES file to the IGES file generated by the vendor's preprocessor.

There is a need for a tool (Tool #5) to load the contents of each file in a neutral database then verify the database contents against the IGES subset information model to arrive at what happened to each entity during the IGES postprocessing and preprocessing functions.

Since the preprocessor test results are already known from task (f) in Section 2.2.2 above, the tool will deduce what happened to the entities during the postprocessing function alone and generate their test results. If the preprocessor was not used in this test, it would have been necessary to directly access the vendor's database to compare its contents with those in the reference IGES file. That may mean the creation of a special preprocessor to access the database.

The test results classify the translation outcome for each entity; examples of these classes are:

- Full functionality, entity same
- Full functionality, entity/entities different
- Partial functionality
- No functionality
- Entity not processed

Test results are generated in a report form and a file form. The file containing the test results will be used for analyzing the data exchange between two systems as shown in Section 2.3 below.

2.3 Analyze Data Exchange between Systems A & B

The purpose of this function is to predict what would happen to entities belonging to a subset of IGES when they are exchanged between two systems A and B. The prediction includes the exchange directions A to B and B to A.

A tool (Tool #6) is needed for this function. The tool will use the test results files for the A and B preprocessors and postprocessors generated by the vendor's IGES translator test functions (Figure 1).

3. DESCRIPTION OF REQUIRED TOOLS

This section describes the functionality of each of the six required tools identified in Section 2.

3.1 Tool #1 (To Upgrade IGES Library to Current Version)

This tool upgrades the IGES test library to the current IGES version. It can be used for version 3.0 upgrades and for future versions of IGES. The upgrades are automatic.

3.2 Tool #2 (To Generate Reference IGES Files)

This tool generates reference IGES files by combining entity IGES files from a library into one file or deleting entities from IGES files to suit a given IGES subset.

3.3 Tool #3 (To Generate Vendor's Script)

This tool generates input script for a vendor's system. The script can be used to create a model in the vendor's database informationally equivalent to the model represented by the reference IGES file.

3.4 Tool #4 (To Verify IGES File Syntax)

This tool verifys the syntax of an IGES file and produces verification results in report and file forms. This tool is not included in the proposed work to be done because there are commercial products which claim the ability to do such verification.

3.5 Tool #5 (To Compare Two IGES Models)

This tool compares two models represented by two IGES files and detects their differences. In the case of preprocessor testing, the comparison identifys what happened to the entities during preprocessing. In the case of postprocessing testing, the comparison results are used with the preprocessor results to identify what happened to the entities during the postprocessing. The comparison results are generated in report and file forms.

Tool #5 has two levels; a basic level and an extended level. The basic level covers all entities except splines and surfaces which would require geometric computation to check if entity deviation is within tolerances. The basic level will output results in report form only. The deduction of postprocessor functionality results will be done through human interpretation of both the preprocessor and postprocessor test reports.

The extended level covers all current IGES entities and will output results in a form which is computer comprehensible. The extended level will automatically produce the postprocessor functionality results. Results from the extended level will be input to Tool #6 for conducting the analysis function. The extended level represents the output IGES information model in terms of the input IGES information model plus delta information about the changes which took place during the translation. This type of representation would make it easy to interpret by a computer program.

The IGES Model Comparison System (IMCOS) developed at the University of Karlsruhe, Germany, is an attempt to do the functions covered by the basic level of Tool #5. However, IMCOS is not designed to be extensible to cover the extended level of Tool #5 and the consequent support for Tool #6. If IMCOS is used as the basic level of Tool #5, significant effort will be needed to do the extended level of Tool #5 using a different base software.

3.6 Tool #6 (To Predict Data Exchange Results for Two Systems)

This tool analyzes the preprocessor and postprocessor test results for two vendor systems A and B in order to predict what happens to entities when the data is exchanged from A to B or from B to A. Tool #6 is dependent on the availability of the extended level of Tool #5.

4. PROPOSED STATEMENT OF WORK

CRD is proposing a two-project effort to cover the development of tools required for the IGES translator verification process. A description of these tools is included in section 3.

The first project is the primary project in which most of the testing tools will be developed. The second project complements the first one and focuses on developing one of the tools aimed at automating the generation of test models in the vendor's system. Since the success of this tool is dependent on the ability to run the vendor's system using a script file or equivalent, a proof of concept is essential; accordingly, a separate project is proposed for achieving that objective.

Each of the development projects and phases is targeted to be a self contained package serving a specific purpose. Any dependencies between projects or phases will be stated below.

Any software developed as part of this project will be placed in the public domain. Tools developed in this project will require the use of internal CRD software developed outside the project. CRD will place the source code of this internal software in the public domain. CRD will not provide support for the code after the end of the proposed projects.

4.1 PROJECT (A): DEVELOPMENT & VALIDATION OF TESTING TOOLS

Project Objectives

The main objective of this project is to develop four testing tools and validate them within a basic testing system that includes the script generator tool, which is developed in Project (B). Another objective is to prove the transportability of the developed code by porting it to IBM[1] hardware.

Project Scope

The scope covers the development of Tools #1, #2, #5, and #6. It also covers the delivery of two versions of the code; one for the DEC[2] VAX[3] hardware, and one for the IBM 43XX or 3XXX hardware. This project consists of the following three phases:

PHASE I: DEVELOP TESTING TOOLS

Phase I Objective

The objective of this phase is to develop tools needed for the vendor's IGES preprocessor and postprocessor testing.

Phase I Scope

This phase includes the development of Tool #1, #2, and the basic level of Tool #5. Tool #4 is assumed to be commercially available.

Tool #1 is used for upgrading IGES test files from earlier versions to the current version. Tool #2 is needed for both preprocessor and postprocessor testing since it generates the IGES reference files. Tool #5 is used in both tests to generate test results.

It is assumed that an IGES subset information model exists. Tools #2 and #5 use that model to control their functions.

[1]IBM is a trademark of International Business Machines
[2]DEC is a trademark of Digital Equipment Corporation
[3]VAX is a trademark of Digital Equipment Corporation

Phase I consists of the following four tasks:

Task A-1 Develop File Upgrade Tool #1

This task provides a tool for upgrading the IGES test library files to IGES version 3.0 and for future library upgrades.

CRD will upgrade some files in the current IGES test library to version 3.0 to demonstrate the functionality of the tool. It is assumed that the principal work involves reformatting issues such as addition of the version number in the global section. Two specific corrections are anticipated and will be performed. The first is the correction, where the error occurs, of the misunderstanding which caused the subordinate entities to some annotation entities to point to the same transformation matrix entity as their parent. The second is the determination and correction of the subordinate entity flag according to whether the entity is physically dependent, logically dependent, or independent.

Files are put into the General Electric Neutral Database (NDB) using the IGES version 2.0 translator and then processed to correct the two known abnormalities. Then data is output to an IGES file using IGES version 3.0 translator.

The software used for the version 3.0 upgrade can be used for future upgrades. A program will be needed to take care of differences introduced by the new version.

Task A-2 Develop Reference File Generator Tool #2

This task consists of developing a tool for generating reference IGES files conforming to a given subset of IGES defined by an IGES information model. This tool will be able to generate the file using two modes. The first mode is merging existing entity IGES files into one file. The second mode is deleting entities from existing IGES files to generate the reference file. Both modes may be used to build the file depending on the available IGES files. Current IGES test library files will be used to demonstrate this tool.

Task A-3 Develop Comparison Tool #5 (Basic Level)

The basic level of Tool #5 is developed in this task. The extended level is covered by Task A-6 in Phase III. The basic level compares two models represented by two IGES files and detects their differences. It covers all entities except splines and surfaces. The basic level will output results in report form only. The deduction of postprocessor functionality will be done through human interpretation of both the preprocessor and postprocessor reports.

Task A-4 Validate Basic Testing System

This task consists of validating Tools #1, #2, and #5 (basic level) developed in this Phase in addition to their use as part of a basic testing system that includes Tool #3 which is developed in project (B). A demonstration of all these tools working together is the main outcome of this task.

PHASE II: PROVE CODE TRANSPORTABILITY

Phase II Objective

The objective of this Phase is to demonstrate that the code developed or used in Phase I is operational on a machine that is different of the development machine.

Phase II Scope

The development of the code will be on DEC VAX hardware. To prove the code transportability, an equivalent version of the code will be prepared and its functionality will be demonstrated on IBM 43XX or 3XXX hardware.

This Phase consists of the following task:

Task A-5 Port Code to IBM Hardware

The DEC VAX version of the code will be analyzed and necessary modifications will be made to make it run on IBM 43XX or 3XXX hardware. The modifications and the reasons for doing them will be documented. The main output of this task is demonstrating that the IBM version is operational. This effort can be used as a guide for porting the code to any other type of hardware.

PHASE III: DEVELOP EXTENDED COMPARISON & ANALYSIS TOOLS

Phase III Objective

The objective of this Phase is to develop an analysis tool for predicting the results of data exchange between two systems. This phase will also extend the tool which compares two IGES models to cover all the current IGES entities and produce computer comprehensible test results.

Phase III Scope

The analysis tool (Tool #6) is dependent on the completion of Tool #5. Accordingly, this phase covers first an extension of the basic level of Tool #5 to cover all current IGES entities and to be capable of generating test results in a file form comprehensible by Tool #6. The analysis tool will provide the user with detailed prediction of what happens to each entity passed between two systems via IGES. This tool can be expanded in the future to analyze data exchange through a series of systems.

This Phase consists of the following four tasks:

Task A-6 Develop Comparison Tool #5 (Extended Level)

This task builds on the basic level of Tool #5 to produce an extended level. The extended level of Tool #5 covers all current IGES entities and will output results in a form which is computer comprehensible. This level will carry out geometric computation for spline and surface entities to check if entity deviation is within tolerances. This level will automatically produce the postprocessor functionality results. Results from Tool #5 will be the input to Tool #6 for conducting the analysis function.

The extended level represents the output IGES information model in terms of the input IGES information model plus delta information about the changes that took place during the translation. This type of representation would make it easy to interpret by a computer program.

Task A-7 Develop Analysis Tool #6

This task covers the development of Tool #6 which can be used for analyzing the preprocessor and postprocessor test results for two vendor systems A and B in order to predict what happens to entities when the data is exchanged from A to B or from B to A. For Tool #6 to operate, Tool #5 (extended level) must be available.

Task A-8 Validate Extended Testing System

This task covers the validation of Tool #5 (extended level) and Tool #6 as part of an extension of the basic system validated in Task A-4 in Phase I.

Task A-9 Prove Code Transportability

This task consists of preparing a version of the code developed in tasks A-6 and A-7 which would operate on the IBM 43XX or 3XXX hardware. The task includes demonstrating that the code is operational on that IBM hardware.

Project Deliverables

CRD will deliver the following:

- Source code for the NDB plus its updated IGES preprocessor and postprocessor.
- Source code for an enhanced version of the Common Database Interface Language (CDIL) compiler and the runtime interface code necessary to run a CDIL-compiled program against the NDB. There is a CDIL version placed in the public domain by the U.S. Military Academy. The deliverable code mentioned above is an enhancement of that version.
- Source code for the CDIL programs developed for the Version 3.0 upgrade.
- At least five IGES test files upgraded to IGES Version 3.0 as a proof of Tool #1 capability.
- Tool#2 source code for generating IGES reference files from existing IGES files. For any given IGES subset, this code will generate a compatible IGES file.
- Tool #5 (basic Level) source code for comparing two IGES models represented by the reference IGES file and the vendor's output IGES file. The program will generate vendor's preprocessor and postprocessor test results in a report form. Splines and surfaces are not included in this basic level.
- Source code extending Tool #5 (basic level) code to cover all current IGES entities and to output test results in a computer comprehensible form.
- Tool #6 source code for predicting the results of exchanging data belonging to an IGES subset between two CAD/CAM systems. The predictions are based on the preprocessor and postprocessor test results produced by the extended level of Tool #5.
- The above software will be delivered in two versions. One for the DEC VAX hardware and the other for the IBM 43XX or 3XXX hardware.
- Documents describing the delivered code, its installation, and usage.

4.2 PROJECT (B) : SCRIPT GENERATOR DEVELOPMENT & PROOF OF CONCEPT

Project Objectives

The main objective of this project is to develop a script generator tool and proof its concept by applying it to two different vendor systems. Another objective is to demonstrate that the code is operational on DEC and IBM hardware.

Project Scope

The scope covers the development of Tool #3, the CDIL code needed for its use with the Calma DDM[1] and CADAM[2] systems, and proving that it works on DEC VAX and IBM 43XX or 3XXX hardware.

Task B-1 Develop Script Generator Tool #3

This task covers the development of a tool which can generate an input script file for a vendor's system based on a reference IGES file. The script file can be used to automatically create a model in the vendor's database equivalent to the model represented by the reference IGES file. The code developed in this task is a module common to all vendor's systems. In order to be able to use the tool, a CDIL program is needed for each vendor's system to take care of entity mapping.

Task B-2 Develop Programs for Use of Tool #3 with Two CAD Systems

This task covers the development of CDIL programs needed for the use of Tool #3 with the Calma DDM and CADAM systems. In the case of CADAM, an equivalent to a script file will be the program's output.

Task B-3 Validate Tool #3 and Compare Its Use with Manual Methods

In this task, Tool #3 and the two CDIL programs will be validated and the results are compared with the case when the vendor's models are created manually. The objective of the comparison is to prove the validity of the concept of Tool #3.

Task B-4 Prove Code Transportability

This task consists of preparing and demonstrating a version of the code, developed in tasks B-1 and B-2, which would operate on the IBM 43XX or 3XXX hardware.

Project Deliverables

CRD will deliver the following:

- Tool #3 source code suitable for generating a script file from a standard reference IGES file. The software consists of a special output module from CDIL designed to write script files for a variety of CAD/CAM systems.
- CDIL source programs to generate script writing programs for two CAD/CAM systems using Tool #3. These two systems are the Calma DDM the CADAM systems.
- The above software will be delivered in two versions. One for the DEC VAX hardware and the other for the IBM 43XX or 3XXX hardware.
- Documents describing the delivered code, its installation, and usage.

[1]DDM is a trademark of the Calma Company
[2]CADAM is a trademark of CADAM, Inc.

4.3 SCHEDULE

Project (A)

The target start and completion dates for each of the project tasks are as follows:

TARGET DATES IN MONTHS ARO			
PHASE	TASK	START	COMPLETE
I	A-1	0	3
I	A-2	0	10
I	A-3	0	10
I	A-4	10	12
II	A-5*	11	14
III	A-6	10	18
III	A-7	17.5	22
III	A-8	22	23.5
III	A-9	23	24

Project (B)

The target start and completion dates for each of the project tasks are as follows:

TARGET DATES IN MONTHS ARO		
TASK	START	COMPLETE
B-1	0	10
B-2*	7.5	10.5
B-3	10.5	12.5
B-4	13	14

* Preliminary estimate; may change when the IBM environment is analyzed

APPENDIX E
****

PLAN FOR

RESEARCH AND IMPLEMENTATION TESTING

IN

PRODUCT DATA EXCHANGE

21 October 1986

Bradford Smith
National Bureau of Standards
A353 Bldg 220
Gaithersburg, MD 20899
(301) 921-3691

1 DEVELOP A COMPREHENSIVE TESTING METHODOLOGY

GOAL:

A rigorous methodology for testing implementations of the IGES Specification is the key to developing quality translators for widespread production use of IGES. This task will develop the technology and the tools to adequately test IGES file translation software. It will also devise and document the criteria to be used to evaluate the degree of success attained. In addition to the formal methodology, actual testing requires an entity subset specification and a suite of documented test cases. Application subsets and test cases will be developed by other tasks of this plan.

APPROACH:

The testing methodology must be based on definitive procedures and a categorization of the problem into workable segments. A careful and complete identification is needed of all potential error sources or barriers to an intersystem data exchange. Categories of IGES entity types needed for the exchange of specified user data sets (i.e., 2D Drafting or 3D Wireframe Models) must be developed. The methodology will include both a comprehensive test plan for gathering data and mechanisms for analysis of collected data.

Software tools must be specified and developed to complement the methodology. The tools will be in the public domain and will be constructed so as to be easily agumented as new IGES entities are published. The software programs are needed for preparing, editing and documenting both the IGES test files and the scripts needed to direct CAD operators in creating the proper models. Additional programs are needed to analyze files for proper data organization and content. Finally, software is needed for the analysis of test results to determine whether a CAD system's translators have conformed to test objectives.

Policies must be established for the documentation of entity mappings, application subsets and test cases. Finally, formats and procedures are needed for feedback of information to IGES translator implementors and for presentation of testing results to the public.

TARGET MILESTONES:

A Publish a generic plan for gathering data, testing translators, determining success, returning comments to vendors, and communicating the results to the CAD/CAM community.

B Develop methods for analyzing the data collected, including mechanisms for comparison and presentation.

C Develop a methodology for presenting and using vendor specific IGES entity mappings to predict degree of exchange completeness among dissimilar CAD systems.

D Document entity subset concept and guidelines for developing and using generic or applications subsets.

E Publish a users guide for testing and validation of IGES translators.

F Initiate the development of a rigorous method for comprehensive testing of both pre- and post-processor translators. Identify software tools needed.

G Publish the criteria for an acceptable IGES translator verification program.

H Document the applications area entity subset concept and provide guidelines for users to perform end-to-end certification testing.

I Develop documentation standards for test cases and procedures for test case validation.

J Develop specific test methods for error checking, system limit testing, and numerical error propagation.

K Develop a program to provide generalized plotting directly from an IGES file.

L Develop a utility program to check conformance to a published subset of IGES entity types.

M Develop validation techniques for solids model data exchanges.

2 IMPLEMENT A TRANSLATOR VERIFICATION PROGRAM

GOAL:

Assuring a good quality of data exchange through the neutral IGES format requires careful and complete testing of the involved translators. This can only be attained through a comprehensive program of testing, the cost of which is beyond all but the largest of user companies. An independent, self-sustaining, centrally run verification program utilizing the testing methodologies approved by the IGES Organization seems to offer the only rational solution.

APPROACH:

This effort will initiate a national translator verification program by providing the resource material and the startup consulting support. Criteria will also be developed for measuring the effectiveness of the verification program. Theverification program makes use of deliverables from all the other tasks of this plan. While the goal is to develop a centralized testing facility, it should be noted that the test procedures are generic and can be performed by any individual at any site. This task provides the startup resources and the consulting needed by a national verification program until it can become self-sustaining.

TARGET MILESTONES:

A Define the relationship between the testing program and the IGES Organization.

B Develop the criteria for ranking, rating or accepting a candidate in the testing process.

C Document a format for presentation of results.

D Develop a policy and procedures guide for a national verification program.

E Initiate a National IGES Translator Verification Program run by a neutral third party.

F Provide consultation and startup resources.

G Review and audit program one year after initiation.

3 DEVELOP ENTITY TEST CASES GOAL:

A wide range of verified correct test cases must be developed as a basis for individual translator testing and for intersystem testing at user or vendor sites. The range should cover a wide spectrum of entity types, user applications, model scales, file sizes, and data organization methods. Test Library cases serve the needs of software testing and provide examples of how IGES-defined entities are to be used to express various part model data.

APPROACH:

Version 1.3 of the Test Library provides test cases for IGES 1.0 entity types. Additional test files will be created to represent current versions and various application areas of IGES, to represent multiple entity relationships, or to show different schemes for data organization within an IGES file. Finally, a series of test cases will be developed to intentionally stress system limits and to check postprocessors for error detection capability.

All files generated will be suitably documented and thoroughly checked for conformance with the applicable IGES version. Documentation will include the purpose of the test, the entity content, a script for generating the test case, the information to be carried by each entity and the evaluation criteria for measuring the success of the test.

TARGET MILESTONES:

A Revise Version 1.3 test cases and develop new cases for all IGES entity types. Publish the document.

B Gather variety of benchmark test cases, analyze them for conformance to the Specification and document them for use.

C From a single test case script, generate an IGES file on each of a variety of CAD systems to form a suite of "flavor" test cases.

D Develop test cases to stress system limits.

4 DEFINE APPLICATIONS AREA SUBSETS

GOAL:

Each new applications area in IGES will require a definition of information content through formal modeling techniques, a specification of how this information is unambiguously mapped into or represented by the IGES entities and a definition of the IGES entity subset for the application. Also needed will be a series of test cases, a method for organizing the data within the IGES file and demonstrations of intersystem exchange.

APPROACH:

Much new technical work has been accomplished since the publication of Version 2.0 and is included in the Version 3.0 document. A substantial portion of this new capability is aimed at providing extensions for electrical, drafting, finite element, and plant design applications. It is necessary for this work to be implemented, tested, and, if necessary, modified or extended to satisfy objectives for each application area.

For each new application capability in Version 3.0, vendors and users should be identified for trial implementations. Entity subsets, applications guidelines, information mappings and sample test files should be made up of typical engineering uses of this data. Errors, omissions, or extensions should be documented where found and submitted for consideration.

TARGET MILESTONES:

A Define initial candidates for applications entity sets.

B Document current practice and procurement specifications on applications entity sets.

C Gather, analyze and document existing applications test cases.

D Document any existing compliance specifications and applications guidelines.

E Develop a defined subset of IGES entity types for application to engineering drawings.

F Document and test a preliminary suite of IGES benchmark files to demonstrate drafting applications.

G Develop a defined subset of IGES entity types for application to 3D mechanical product models.

H Document and test a suite of IGES benchmark files to demonstrate 3D mechanical applications.

I Develop a defined subset of IGES entity types for application to electronic printed wiring boards.

J Document and test a suite of IGES benchmark files to demonstrate electronic product applications.

K Develop a defined subset of IGES entity types for application to finite element mesh models and their associated engineering properties.

L Document and test a suite of IGES benchmark files to demonstrate finite element analysis applications.

M Develop a defined subset of IGES entity types for application to archetectural engineering and construction.

N Document and test a suite of IGES benchmark files to demonstrate AEC applications.

5 DEVELOP PRODUCTION WORTHY TRANSLATORS

GOAL:

An intersystem data exchange is basically limited by the matching of functionality of the two systems and the quality of the pre- and post-processor translators available for use. A necessary condition is that both translators be capable of processing all information to be transferred. For a production-worthy exchange, this processing must be complete and accurate. Where problems are encountered, the translator software must effectively deal with errors and supply sufficient information to the CAD operator to detail the nature of the problem. Hence, the quality of data exchange rests on the correctness and the completeness of translator implementation.

It is recognized that writing translators is a function usually performed by each vendor's staff. Little material other than the IGES Specification is available to assist the writer. This task will develop necessary resource material, will generate the user pressure for compliance and will feed back to the implementor the results of analysis and intersystem testing on his translators.

APPROACH:

A first step in assessing whether a data exchange will be successful is a comparison of entity mappings between the translators involved and the anticipated content of the data file. Information must be made available on translator entity capability and the way an internal entity is mapped in and out of IGES. User guides are needed on how to plan for a successful and complete data exchange and for the many considerations involved in writing an IGES translator. Finally, mechanisms will be created for making the resource material freely available.

TARGET MILESTONES:

A Request and document vendor entity mappings.

B Develop a schedule of entities to be tested.

C Publish a User Guide to IGES Data Exchange.

D Plan at least one major public demo each year.

E Publish sample specifications for procurement of IGES translators and IGES data files.

F Develop a User Guide for writing IGES translators.

G Publish Recommended Practices for error checking and reporting by translator software.

H Provide error feedback from testing to vendors.

6 PERFORM EXTENSIVE INTERSYSTEM TESTING

GOAL:

Implementors typically develop their IGES translators based on the printed specification and tend to work at arm's length from each other with little opportunity for testing with other vendor systems. This approach virtually assures that differing interpretations, or flavors, of the IGES Specification will be promulgated. As this software is released to users, implementors are warry of making changes, feeling an obligation to their customer base to maintain the utility of any previously produced datasets. Hence, an active and focused central program of intersystem testing is essential in order to develop smooth and capable intersystem exchange capability across a broad range of implementations.

APPROACH:

The intersystem testing will provide close coupling between the needs of users and the capabilities of vendor implementations in a data exchange. The barriers to complete data exchange will be analyzed and documented. Policies and software will be developed to alleviate the problems where possible. This task requires the active cooperation of implementors and the outputs from other tasks in this plan. Results of the frequent intersystem testing will be carefully evaluated to track the level of implementation and to refine the testing methodology, the test cases, the software tools and the other testing materials.

A vendor's view of IGES tends to be limited to mapping his data structure to and from a specified set of IGES entity types that he has implemented. Users on the other hand are interested in the more demanding task of an end-to-end exchange of the data structure.

This task is designed to develop competence in the full exchange process, to understand the limits of the technology and to assess the quality and completeness of the testing methodology, the application area entity subsets, the software tools and the test cases developed by the other tasks. Through extensive intersystem testing, guidelines can be developed for the user community to maximize the information content capable of being exchanged or archived in IGES.

TARGET MILESTONES:

A Assess the completeness of the IGES Testing Methodology.

B Validate all test cases generated.

C Verify reported entity mappings.

D Test developed software tools and refine if necessary.

E Develop database of implementation capability.

F Document any inherent problems of CAD database exchange and recommend solutions.

G Validate applications area entity subsets and the information mappings.

7 DEFINE REQUIREMENTS FOR ARCHIVAL QUALITY DATA

GOAL:

The experience with IGES to date has been for exchange of datasets. Success has been attained only through careful planning and testing of the data paths used and the close collaboration of sender and receiver. However, the need also exists for the archiving of product data. This is a much more demanding activity. The data is stored at one point in time and transferred to a receiving system at a some relatively unknown future point in time. The receiver acting alone must resolve any problems that are encountered.

APPROACH:

This task will develop a full understanding of the "archive" problem and will result in detailed guidelines and software tools for ensuring the integrity of datasets before they are accepted into archival storage.

TARGET MILESTONES:

A Host a workshop on the archiving problem.

B Develop documentation profile about the IGES file including the CAD system which generated the file, data organization method and information content.

C Determine if above information should be placed in START Section or if additional data elements are needed in the GLOBAL Section.

D Publish a report on file flavoring with emphasis on problems for archiving and suggesting solution approaches.

E Document procedures for archiving IGES data.

F Develop software tools needed to check files before archiving.

8 DEVELOP PDES VERSION 1.0

GOAL:

PDES Version 1.0 will provide entity capability for mechanical piece parts, mechanical assemblies, electrical printed wiring board products including both schematic and physical designs, AEC models, FEM models and drafting applications. Supporting technical areas will also be provided in PDES for manufacturing, solids modeling ,curve and surface modeling and presentation data.

APPROACH:

The development of Version 1.0 is intimately tied to the PDES Initiation Activities described earlier. Major inputs have come from the Air Force PDDI contract, and much is expected from the evolving GMAP contract. Details of approach are documented in a draft plan prepared by the PDES Project Manager.

TARGET MILESTONES:

A Develop a detailed project plan.

B Identify functional content of PDES Version 1.0

C Prepare updated and enhanced project plan.

D Choose formal data modeling methodology for PDES.

E Prepare data models for each applications area.

F Develop conceptual schema for PDES information.

G Specify a data specification language as the interface between the logical layer and the physical layer.

H Develop physical file structure.

I Embed the conceptual schema in the physical file structure.

J Publish initial draft PDES Version 1.0

K Publish final draft PDES Version 1.0 for practical test and validation

L Develop algorithms to convert data in then current IGES Version to PDES Version 1.0

M Release PDES Version 1.0

N Release ANSI Standard for PDES

9 DEVELOP TOOLS FOR MAINTAINING THE INFORMATION MODELS

GOAL:

To be useful to the IGES community; a significant portion of the standard, the conceptual schema and its mappings, must exist in computer readable form. The information model will exist in several forms during the life cycle of a particular change to the standard. For example, the application committee interested in the change is likely to utilize IDEF1X to study and modify a proposed change. Later, the logical layer committee will use IA to complete the analysis of the change and map it into the conceptual schema. The physical format committee will use DSL to complete the mapping of the change into the physical format.

APPROACH:

Each of these forms of the information model should be recorded in a single format and language. Where automatic conversion between different forms is possible, software for accomplishing the conversion should be developed and made available to committee members. The software must read and write the standard format for recording the model.

TARGET MILESTONES:

A Modeling methodology committee agree on set of tools necessary to do the job.

B Modeling methodology and physical format committee define standard format (probably an extension to DSL).

C Software committee prepare specifications for any required software modules.

D Publish conceptual schema in standard format and make available to interested members.

APPENDIX F
****

COMPLETENESS OF PRODUCT DATA EXCHANGE

A conceptual model for product definition information exchange is presented along with a discussion of the responsibilities of the system users in effecting a successful exchange. Also presented is a discussion of the role of the IGES Organization in supporting these users in their quest for accurate and reliable information exchange on a routine basis.

< < < INTRODUCTION > > >

Computer aided design and drafting (CADD) systems are becoming quite commonplace in many segments of our society. The proliferation of traditional CADD systems, and the introduction of low-cost, microcomputer-based CADD systems have given many people and organizations their first taste of the creation and dissemination of product definition information in electronic form. This increase in the use of CADD and other systems which generate product definition data is causing more and more people to become concerned with the exchange of information between systems which are incompatible. It is the desire of these people to establish accurate and reliable exchange of product definition information between dissimilar systems on a regular and frequent basis. In this paper, the author refers primarily to the kind of data used on CADD systems but the concepts are equally applicable to any of the systems which deal with product definition data.

In the following discussion, it is important to distinguish between data and information. Data is defined here as simply a collection of facts. In contrast, information is made up of data as well as the relationships among the data items and is, therefore, more meaningful and valuable.

< < < A CONCEPTUAL MODEL FOR INFORMATION EXCHANGE > > >

The conceptual model for product information exchange used here is a hierarchical one consisting of four layers (Figure 1). Starting at the highest layer, these are (1) a common knowledge base with its definition of information content, (2) representational models with their standards and conventions, (3) data translation standards and conventions, and (4) data transmission standards and conventions.

In any exchange of information, there needs to be a common understanding of certain fundamental principles between the parties to the exchange. This common knowledge base is necessary so that the meanings of what is said can be understood. An important part of this common knowledge base is an understanding of the basic information which is required to communicate some meaning from one party to another. As an example of this, consider the exchange of drilled hole information between a designer and an N/C programmer. Both need to understand certain basic concepts for drilling holes; they also have to define what information is needed in order to specify the holes to be drilled (e.g., location, depth, diameter, tolerances on each of these lengths,

etc.).

The representational models concern themselves with how the information content of the knowledge base is to be represented in a particular CADD database. Standards and convetions need to be developed so that consistent representation will be found within a given system, but the representational models may not be, and do not have to be, the same between systems, especially if the systems are supporting different disciplines. For example, the representation of a drilled hole on an engineering design system does not have to be the same as its representation on an N/C programming system, so long as the inforamtion content is present on both systems. In this example, the engineering design system might represent a drilled hole by two circles connected by three lines with yet a fourth line indicating the angle of the drill at the bottom of the hole, while the N/C programming system might represent it as two points (at the beginning and end of drill penetration) and a circle.

Another example of representation conventions in a machining environment could be that external cast features are to be found on layer 1, internal machined features on layer 10, through holes on layer 50, etc. In an electrical environment, the representation conventions could be that the outline of the printed circuit board is to be found on layer 20, the physical details of each of the n layers of the circuit are to be found on layers 30+n, and the supporting design details (e.g., schematics and netlists) are to be found elsewhere on specific layers.

Obviously, these sorts of conventions are highly dependent on the parties involved in the information exchange. Even with the attempts at standardization of such industry groups as the Automotive Industry Action Group (AIAG) and the Electronics Industry Association (EIA), there is still much room for establishing unique conventions between the parties at each end of an information exchange.

Frequently, the term information modelling is used to mean the combination of the information content and the associated representational model.

Data translation standards and conventions address how the data is transformed from the native form of one system to the native form of another system. There are two fundamental approaches to this task; direct translation and indirect translation. Direct translators take the data from the sending system and transform it immediately into the equivalent form on the receiving system. Direct translators have to know the internal data structures and data definitions of both systems involved in the exchange. Indirect translators transform data between the native form on one system and an intermediate, standard form (usually referred to as a neutral form). Indirect translation is a two-step process: translating from the native form on the sending system into the neutral form, and translating from the neutral form to the native form on the receiving system. IGES (the Initial Graphics Exchange Specification) is an example of an indirect translator.

Data transmission standards and conventions define the process of moving the data from one system to another. This layer of the exchange model is concerned with the physical media and the standards and conventions which apply to writing data to a medium and reading the data from it. For example, the most common means of transmitting data in IGES form is by way of magnetic tape. By convention of the IGES community, the tapes are written at 1600 bits

THE CADD INFORMATION EXCHANGE ENVIRONMENT

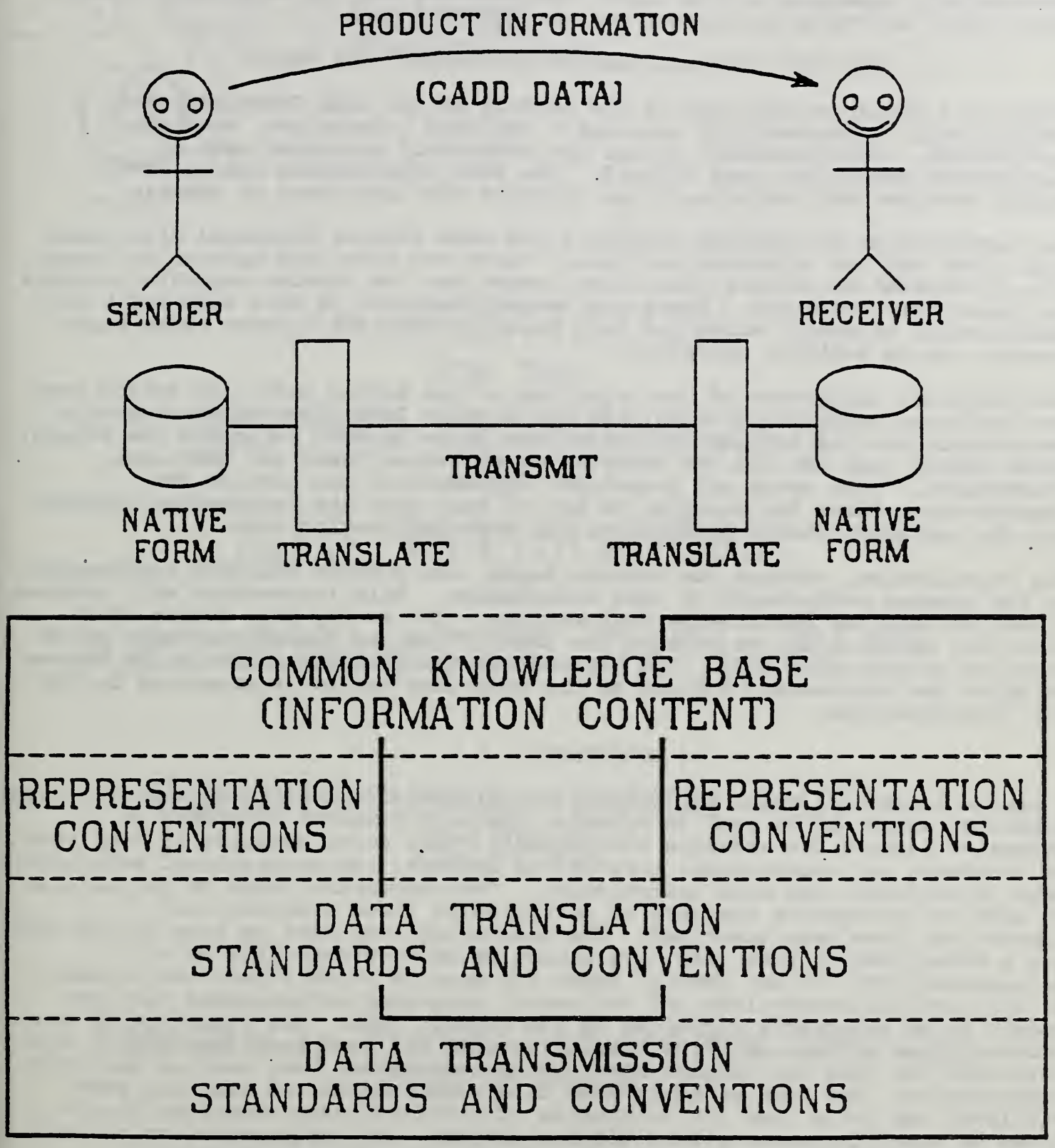


FIGURE 2

To test the information modelling levels, end-to-end tests must be run between the systems which sit on each end of the information exchange. These are the tests which will ultimately tell whether the exchange can work. These tests are entirely dependent on the users' environments and the underlying systems. They cannot be run by any agency for the "general case."

< < < HOW THE IGES ORGANIZATION CAN SUPPORT THE USER > > >

There is a desire on the part of the members of the IGES Organization to assist users in successfully creating a reliable information exchange environment. Unfortunately, no one can completely guarantee such an environment except the user himself. The IGES Organization can, however, supply services and tools which can increase the likelihood of success.

The Organization can publish tutorials and case studies concerned with these user tasks and how to accomplish them. There are many IGES members who have already started to wrestle with these issues and can provide valuable insights for those just beginning. There are actual examples, of both successful and unsuccessful attempts, which can help users to progress through these tasks smoothly while avoiding pitfalls.

The Technical Committees of the Organization can define modelling entity sets for different application areas, and can provide IGES files which correctly demonstrate how the application entity set is to be used to create the models. These entity sets can aid the users in establishing their own modelling conventions. Both users and translator implementers can look to the demonstration files for examples of how to deal with the information content and the representational conventions for that application area.

The Organization, through its testing agent, can provide reliable information on the general performance of IGES translators. This information will consist of entity maps and performance designations. The entity maps define which modelling entities on one side of the translation are transformed into which entities on the other side. The performance designations indicate the degree to which the information content of the modelling entity is preserved during the transformation.

< < < CONCLUSION > > >

A simple conceptual model for product definition information exchange has been presented which can be used to organize the work required to create an effective information exchange environment. This model consists of four sets of standards and conventions: information content, representational modelling, data translation, and data transmission. The responsibilities of the parties to such an information exchange in establishing these standards and conventions have been described. The underlying work must be done by the user (or a consultant to that user) and cannot be done successfully by an independent body for the general case; the details which affect the contents of the specific conventions and the users' operating environments vary too widely to be reasonably supported in the general case. The functions of the various types of testing which must be carried out have been described. Also described has been the support role for an independent body such as the IGES Organization. The type of assistance this organization can lend has been outlined, and falls into the categories of providing user education in the areas of modelling conventions, testing methods, and application entity sets; and providing reliable performance information about translators.

APPENDIX G
****

INITIAL
GRAPHICS
EXCHANGE
SPECIFICATION

DOCUMENTATION
FOR THE
TESTING METHODOLOGY
OF
IGES TRANSLATORS
AND
IGES FORMATTED DATA FILES

WORKING PAPER VERSION 0.03
19SEP86

PREPARED BY: TESTING METHODOLOGY COMMITTEE
IGES ORGANIZATION

APPROVED BY: _____

BRADFORD SMITH
CHAIRMAN
IGES STEERING COMMITTEE

0291X

* Preface to the Working Paper *

The enclosed material represents ideas and concepts to be considered by the IGES Technical Committee members. Your comments and ideas are solicited and should be addressed to the editor. As this material is subject to major changes during the technical review, it should not be taken as representing a consensus of any IGES committee.

This working paper of the Testing Methodology document contains the collection of the most recent thoughts of the members of the Testing Methodology Committee. It is my intent to prepare a new version shortly before each of the quarterly IGES meetings, then collect changes and additions at the meetings to incorporate into the next version. In some instances, I will add to or change the working paper to reflect what I thought was the sense of the committee. It is expected that anyone not agreeing with the changes will challenge the new version.

Where items have changed substantially from the previous version, a change bar in the right margin will be found. The working paper version number is made up of a major version number (equal to the version number of the latest release of the document), a decimal point, then the sequence number of the latest version since the last release of the document. Hence, Version 1.07 is the seventh version of the working paper after the first published document version.

J. M. Fleming
Editor
Cummins Engine Co., Inc.
Box 3005, M/C 50142
Columbus, IN 47202-3005

1.0 ABSTRACT:

This document contains the descriptions and explanations of the test methods and test data sets collected and developed by the Testing Methodology Committee of the IGES Organization which is sponsored by the United States National Bureau of Standards. The document also describes the purpose and work of the Committee. This information is intended to be used by people interested in the verification and use of IGES translators.

The test procedures and data sets contained herein are aimed at exercising many aspects of data translation performed as a part of the exchange process for product definition data. Expected results of each test, along with guidelines for evaluating a processor's performance, are also presented.

Suggestions for additions to, and corrections of, this document should be sent to the Chairman of the Testing Methodology Committee via Mr. Bradford Smith, Chairman, IGES Steering Committee, National Bureau of Standards, Gaithersburg, MD 20899

1.1 Acknowledgements

This document is the product of many hours of work on the part of the members of the Testing Methodology Committee. In addition, the following companies contributed employee's time and computer resources to help verify the procedures and test data included.

2.0 INTRODUCTION

2.1 The Charter of the Testing Methodology Committee

The Testing Methodology Committee has been established to create and maintain a collection of test methods, along with their supporting data sets and software tools, aimed at verifying and diagnosing IGES processors and data files. This charge includes coordinating the committee's work with that of other individuals and groups engaged in similar efforts. Development and maintenance work consists of producing at least the specifications which can be used to create and modify the test methods, test specifications (how to run a test), and evaluation criteria (how to judge the results).

2.2 The Scope of IGES Testing

The Testing Methodology Committee should concentrate its efforts on developing and maintaining test procedures which deal with verification testing, validation of processors for specific application subsets, and application-specific, user-performed acceptance testing.

In this context, verification testing is aimed at ensuring that the implementer's claims for entity mapping and functionality of translation are accurate, validation testing is aimed at ensuring that application subsets of IGES/PDES are treated in a manner consistent with the application, and acceptance testing is aimed at ensuring that particular IGES/PDES processors will work adequately in a user's environment.

The immediate goal of the Testing Methodology Committee is to establish a verification program for IGES translators.

An underlying concept to this testing is that of functionality. Functionality of the results of a translation is defined as the degree to which processed part data have been treated as if they were generated on the system being tested.

2.3 The Need for IGES Testing

There are several major reasons why testing IGES processors is necessary. Both implementers and users of the processors are interested in testing. Implementers need a set of widely accepted test procedures and data in order to ensure that their products conform to and adequately support entity subsets for a specific application area of the IGES standard (i.e., verification and validation testing). Users and implementers need a set of widely

2.3 The Need for IGES Testing (cont'd)

accepted test procedures to determine if the processors in question will adequately support the user's data exchange requirements (i.e., acceptance testing). In addition, users frequently desire advice on how to construct test data sets which will accurately reflect these data exchange.

requirements. Finally, since much data exchange, worth millions of dollars, will be accomplished using IGES translators, users are requiring more assurance that the data is not being modified or lost during the exchange process.

2.4 The Fundamental Principles of Testing

A distinction is to be made between demonstrating and testing a product. The objective of a demonstration is to show that the product works (i.e., that the features and capabilities function as advertised). As a result, a data set for demonstration purposes will usually be well-formed and will contain no errors. In contrast, the objective of a test is to expose flaws in the product (i.e., to show where the product does not work as it should). In this regard, data sets for testing should contain purposely malformed and incorrect data.

In his book, The Art of Software Testing {MYER79}, Glenford Myers cites the following basic principles of tests:

- Testing is the process of executing a program with the intent of finding errors.

- A good test case is one that has a high probability of detecting an as-yet undiscovered error.

- A successful test case is one that detects an as-yet undiscovered error.

In the context of generalized IGES processor testing, the test cases developed will be aimed at exposing flaws in important features of the processors. Since the test cases will be known to most of the implementers of these processors, the opportunity will be available to them to construct the software so as to pass the tests. If the contributers to the Test Data Library do their jobs well, this should not be a drawback to the verification process, but a positive influence on the construction of working translators.

3.0 ACCEPTANCE TESTING - A TUTORIAL

This section is provided as an aid to end users who need to determine the suitability of IGES translators for their applications and operating environments. It is impossible for a single agency to provide the level of testing required to make this determination. Therefore, it is up to the end user to examine his/her own environment and to devise appropriate tests to ensure that the proposed translators will perform adequately. A fundamental concept in this work is that of entity mapping. The next section explains this concept. In the sections which follow, then, are explanations of how the end user's acceptance testing should be carried out.

The verification program described later in this document will provide a solid basis for acceptance testing by assuring that the entity mapping claimed by the implementer accurately reflects the processing carried out by the translator, and, in the case of preprocessors, that the IGES entities are correctly formed. Once the end user is aware of the capabilities and limitations of the translators, a more controlled and effective data exchange can take place between dissimilar systems.

The report of the results of the verification testing for a translator plays an important role in assessing how effective the data exchange will be, both into and out of a particular system. This report contains the translator's entity map along with the findings of the verification testers concerning the preservation of functionality and the correctness of the entity formulations.

3.1 Use of Entity Mapping in Data Exchange

In general, entity mapping can be described as the manner in which the implementer of a translator has defined the correspondence between native entity forms (i.e., the entity form maintained within a system) and the IGES entity forms (i.e., the entity form contained in the IGES specification). For example, a string of curve and line segments on a system may be translated into a Composite Curve entity (Type 102) by a preprocessor. This correspondence of a string in the native form to a Composite Curve in the IGES form is the preprocessor's entity mapping for the native string entity. Similarly, a postprocessor may translate a Copious Data entity (Type 106) into a 3D line entity on the receiving system under some circumstances. Thus, the correspondence of the Copious Data entity to the line entity is the postprocessor's entity mapping for the IGES Copious Data entity.

The forms for the preprocessor entity map shall include the names of the IGES entities and the names of their corresponding native entities along with the letter designation which identifies the degree to which the functionality of the IGES entity is preserved in the translation.

The entity maps can be used as a first step in determing how well data can be exchanged between two systems. Knowing the native entities which are to be used on the sending (or initiating) system, the end user can follow each entity through the entity maps for both the preprocessor and the postprocessor to see what the resulting entity will be on the receiving system. The end user can also see from this how good the translation of that entity will be by looking at the corresponding letter designations.

Depending on the end user's application of the data exchanged, imperfections in the mapping and translations may be acceptable. Acceptance tests should be performed to determine the effects of imperfections on the overall data exchange. Based on the results of the acceptance testing, the end user can strive to improve the quality of the data exchange by:

- making use of options provided by the translator,
- adding to and deleting from the list of native entities being used,
- developing intermediate processors to deal with "flavoring" during translation.

Flavoring is a term that is used to describe particular practices built into a translator in situations where, for preprocessors, the underlying system supports entities or relationships which are not directly supported by IGES, and, for postprocessors, the underlying system does not handle specific entities or relationships supported by IGES. Since the manner in which these situations are handled is not standardized, other knowledge than what is available in the IGES document is needed. This additional knowledge is frequently built into the translators.

3.2 Designing an Acceptance Test

To be added

3.3 Evaluating of Accpetance Tests

To be added

3.4 Application Entity Subsets

To be added

4.0 DIAGNOSING ERRORS IN IGES DATA FILES

To be added

5.0 VERIFICATION PROGRAM

5.1 Rationale

It is required that the IGES organization provide a verification process for IGES translators. This requirement is imposed by the ultimate users of these translators - the people whose business requires the exchange of product definition data in electronic form between computer systems.

In devising such a verification, the first consideration is that of deciding what aspects of the translation process can be verified. From the end-users' point of view the ideal verification would be that of a complete, end-to-end data transfer. To do this, however, requires not only a great deal of testing to cover all the necessary combinations of vendors' products, but requires that these combinations be tested for each different user's application and environment. This is an impractical task for a verifying agency.

The following approach to translator verification has been devised to provide users with the greatest amount of reliable information. Also provided is a discussion on how a user is to employ the verification results to establish a reasonable data exchange environment (see Section 3.0).

It is proposed that the thrust of the verification of a processor be the substantiation by an independent agency (the Society of Automotive Engineers--the SAE) of an implementer's claims for the entity mapping and other processing carried out by a translator. The method proposed here is to collect this information from an implementer for each processor to be verified, and then to perform sufficient testing so as to verify that the claims made are correct and that the entity mapping is correctly described.

5.2 Overview of the Verification Process

The verification process is initiated by an implementer (called the Presenter) by filing a Verification Request Package with the IGES Verification Panel of the SAE (give specific citing & address). The Verification Request Package will contain a cover sheet (Figure 5-1), a set of entity mapping forms (Figures 5-2 and 5-3), and any required system and application documentation.

The Verification Request Package sheet contains identifying information about the Presenter and the processor to be tested. It also contains a checklist for the information required to support the verification testing. The documents required to support the testing may vary from system to system. All documents submitted should be listed here.

IGES VERIFICATION REQUEST

PRESENTER NAME: _____

ADDRESS: _____

_____

_____

SYSTEM NAME: _____

RELEASE IDENTIFICATION: _____

RELEASE DATE: _____

HARDWARE CONFIGURATION:

CPU: _____ MEMORY SIZE: _____

SOFTWARE CONFIGURATION:

OPERATING SYSTEM: _____

RELEASE IDENTIFICATION: _____

CAD SOFTWARE: _____

RELEASE IDENTIFICATION: _____

IGES PROCESSOR NAME: _____

RELEASE IDENTIFICATION: _____

RELEASE DATE: _____

DATE: _____

REQUEST FOR VERIFICATION OF:

- ☐ PREPROCESSOR
- ☐ POSTPROCESSOR

IGES SPECIFICATION VERSION: ____

SUPPORTING INFORMATION:

- ☐ PROCESSOR ENTITY MAPS
- ☐ PROCESSOR DOCUMENTATION
- ☐ TAPE ACCESS DOCUMENTATION

OTHER (ITEMIZE):

- ☐ _____
- ☐ _____
- ☐ _____
- ☐ _____
- ☐ _____

SUBMISSION NO.: _____

ASSIGNED TO: _____

DATE: _____

FIGURE 5-1: COVER SHEET FOR THE VERIFICATION REQUEST PACKAGE

PREPROCESSOR ENTITY MAPPING

NATIVE ENTITY	IGES ENTITY	FNCTL CLAIM	TEST REF	FNCTL FOUND	REMARKS (PRESENTER AND TESTER)
	NO.100 CIRCULAR ARC				
	NO.102 COMPOSITE CURVE				
	NO.104 FORM 0 CONIC ARC (GENERAL)				
	NO.104 FORM 1 CONIC ARC (ELLIPSE)				
	NO.104 FORM 2 CONIC ARC (HYPERBOLA)				
	NO.104 FORM 3 CONIC ARC (PARABOLA)				

FIGURE 5-2: PRE-PROCESSOR ENTITY MAPPING FORM

POSTPROCESSOR ENTITY MAPPING

IGES ENTITY	NATIVE ENTITY	FNCTL CLAIM	TEST REF	FNCTL FOUND	REMARKS (PRESENTER AND TESTER)
NO.100 CIRCULAR ARC					
NO.102 COMPOSITE CURVE					
NO.104 FORM 0 CONIC ARC (GENERAL)					
NO.104 FORM 1 CONIC ARC (ELLIPSE)					
NO.104 FORM 2 CONIC ARC (HYPERBOLA)					
NO.104 FORM 3 CONIC ARC (PARABOLA)					

FIGURE 5-3: POST-PROCESSOR ENTITY MAPPING FORM

The Verification Request Entity Mapping Form contains information about the entity mapping between native entities and IGES entities that the translator purports to implement. (Note that there are two sets of forms, one for a preprocessor mapping, and the other for a postprocessor mapping.) The third column on the form gives the Presenter's claim of the functionality of the mapping for each entity listed in the native entity column. See Section 8.1 for an explanation of the functionality designations. This information is necessary to enable the verification testers to determine which test cases need to be executed. The rightmost columns on the form will be used by the testers to record the test results. Presenters may also use the remarks column to identify unusual situations.

After reviewing the Verification Request for completeness, the IGES Verification Panel will schedule the test and assign a tester. Using the information on the request form, the tester will identify the test cases needed from the test library and any that need to be written. The tester will also develop the specific evaluation criteria for the test cases. Finally, the tester will execute the test plan and record the results. These results will be returned to the IGES Verification Panel for a decision on whether the processing of the translator has been verified.

The IGES Verification Panel will notify the Presenter of its findings prior to any public release of the verification material. The Presenter will than have an opportunity (30 days?) to respond to any problems encountered or to appeal the decision of the Panel.

When a translator has been successfully verified, the Verification Package (the request, test results, and summary report) will be forwarded to the NBS for distribution. The NBS will distribute copies of the Summary Report and the Verification Package on request.

If in subsequent use of a verified translator, a user has reason to believe that the translator is not working as verified, the user should notify the Chairman of the IGES Verification Panel (give address of SAE office). Such notification should include sufficient information to recreate the fault.

5.2 Overview of the Verification Process (Cont'd)

The Panel shall forward the notification to the Presenter for response, and to the Test Agency(s) for review. If the Presenter does not adequately resolve the problem, the verification for the translator will be revoked.

5.3 Verification Program

The IGES verification program is being sponsored by the Society of Automotive Engineers (SAE). It is one of several verification programs administered by them. There are four groups which participate in the verification program: the SAE staff, the Technical Board, the IGES Verification Panel, and the testing agency(s)

- Provide administrative services and support.
- Arrange logistics of meetings including meeting facilities.
- Handle invoicing and collection of presentation fees.
- Maintain detailed financial information on incomes and expenditures of the IGES Verification Program. Provide such information to the Technical Board to assist it in establishing appropriate fee structures.
- Arrange for appropriate legal counsel and report the findings and recommendations of legal counsel to the IGES Verification Panel.
- Arrange for appropriate indemnification insurance for the IGES Verification Panel.
- Assure compliance with all prescribed forms and procedures involving the IGES Verification Program
- Hire a capable agency(s) to perform the testing and provide contract administrative support.
- Maintain appropriate records.
- Establish and maintain procedures to ensure the confidentiality of certain results as appropriate.

6.0 OVERVIEW OF TEST METHODS

The testing of IGES processors can be classified in various ways. As discussed in Section 2.2, the entities and parameters to be tested are chosen based on whether the intent of the testing is verification, application validation, or user acceptance. Three levels of test are identified here. These levels are referred to as the physical, logical and functional levels.

On the physical level, the testing is aimed at verifying the ability of processors to handle the syntax of the specifications (e.g., proper recognition of data types). On the logical level, the testing is aimed at verifying the entity mappings defined by the implementers (e.g., a particular entity is translated into and out of the native form in conformance with the IGES specification and in accordance with the implementer's claims). On the functional level, the testing is aimed at determining the degree of equivalence between data models on both ends of a data exchange.

Table 6-1 presents a summary of several key considerations for the different test levels. Details are contained in the following paragraphs.

6.1 Types of Tests

This section presents an overview of a number of different approaches to processor testing. Various combinations of these test types may be used during the testing of an IGES translator. The details of how the test results are evaluated and how success of processing is determined depend on the specific test being run and are to be described in the appropriate testing documentation (see Section 9).

One-way Preprocessor (Figure 6-1) - A test of a preprocessor which translates a known CAD model into a resulting IGES model. This IGES model is then examined to determine the success of the processing.

One-way Postprocessor (Figure 6-2) - A test of a postprocessor which translates a known IGES model into a resulting CAD model. The CAD model is then examined to determine the success of the processing.

Circle (Figure 6-3) - A test of both preprocessor and postprocessor. A known CAD model is translated into an IGES model which is then translated back into a CAD model. The resulting CAD model is compared to the original CAD model to determine the success of the processing. This is sometimes called a Loop Test.

6.1 Types of Tests (Cont'd)

Simple Intersystem (Figure 6-4) - A test of the preprocessor of one system and the postprocessor of another. A known CAD model from one system is translated into an IGES model which is then translated into a CAD model on the other system. The resulting CAD model is compared with the original CAD model to determine the success of the processing. Frequently, variations such as reversing the direction of data flow are performed.

Jury System (Figure 6-5) - A test where a single system under the test is used to run intersystem tests pairwise with a number of other systems. This form of testing is used to determine the suitability of the processors on the system under test in a real-world environment.

Daisy Chain (Figure 6-6) - A test in which a known CAD model is passed through several systems' preprocessors and postprocessors in succession. The resulting CAD models are compared with the original to analyze any degradation of the data caused by the multiple processing.

6.2 Combined Testing

The above methods are combined to test end-to-end data exchange for acceptance testing. Depending on the user's requirements, the test method will be a choice or combination of the simple intersystem test, the jury system test, or the daisy chain test.

Where possible, software tools are being produced to assist in the analysis of the Test Result Models (See Section 9.4).

Table 6-1 Summary of Testing Levels

TYPE OF TEST	Physical	Logical	Functional
WHAT IS TESTED	Syntax and Structure	Entity Mapping and System Limits	End-to-End Data Exchange
BASIS FOR TEST DATA	IGES Specification	IGES Specification and Implementer's Entity Map	Application Requirements and Implementer's Entity Map
TESTER	Implementer	Verifying Agency	End User
TEST METHOD	Preprocessors Modeling script-to-IGES File Postprocessors IGES file-to-database		
EVALUATION	Preprocessors Error Log Postprocessors - Syntax Checker	Preprocessors IGES model comparisons Postprocessors - CAD model comparisons - Operational tests - Geometric verification	Before and after comparison with respect to user criteria - Visual comparison - Operational and functional tests - Database content
PURPOSE OF TEST AND USE OF RESULTS	Error Detection	Formal Verification	Applicability in specific application environment

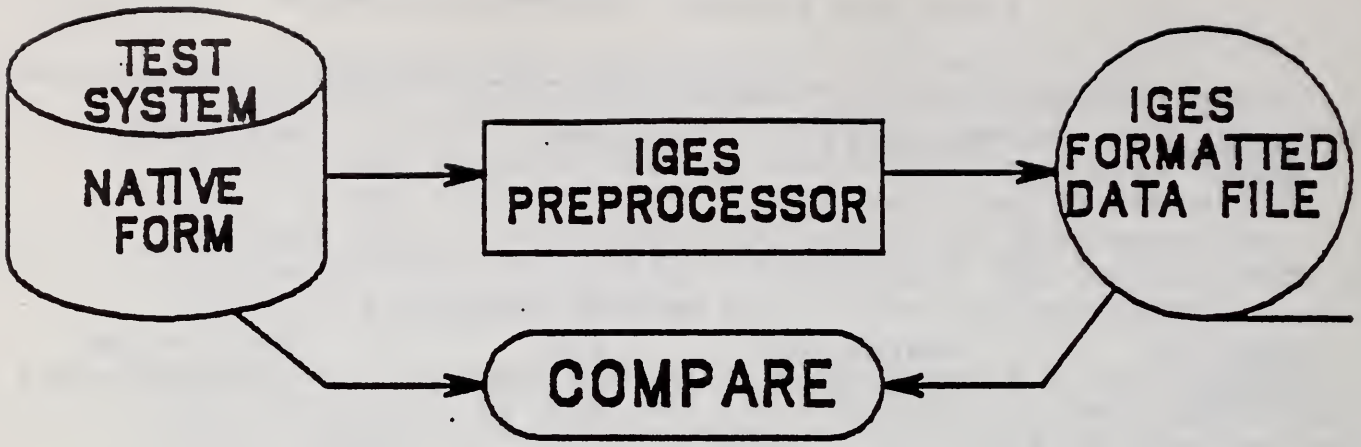


FIG. 6-1: ONE-WAY PREPROCESSOR TEST

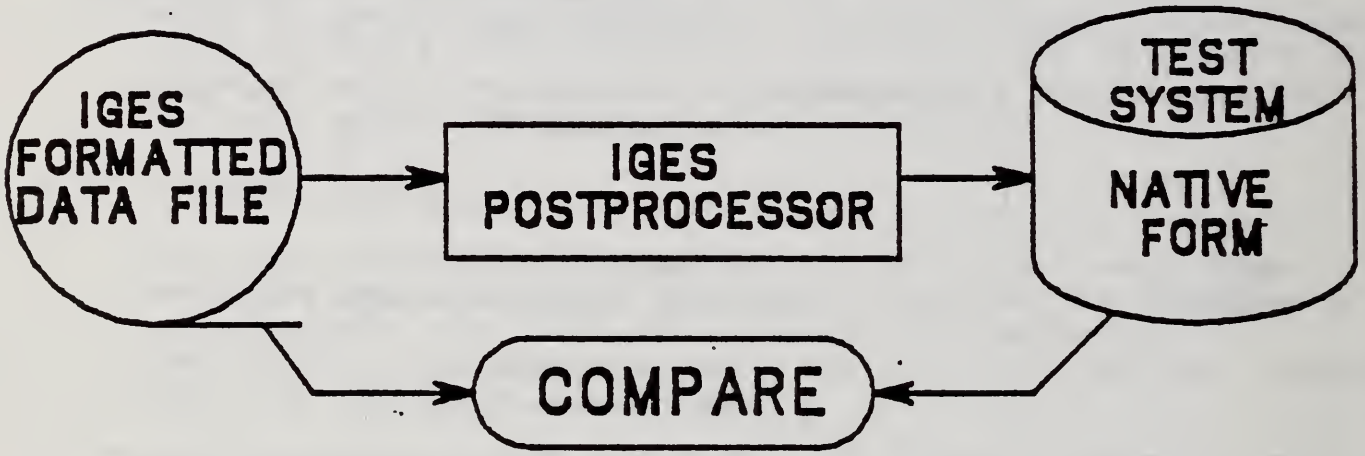


FIG. 6-2: ONE-WAY POSTPROCESSOR TEST

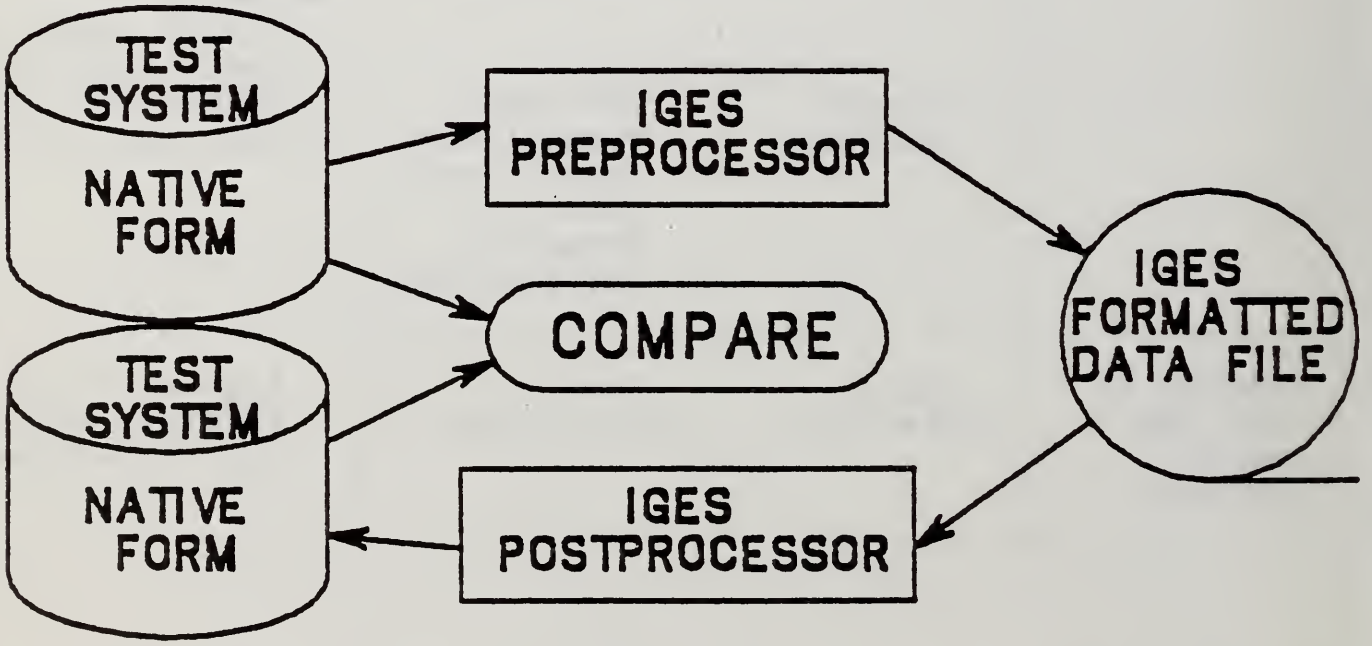


FIG. 6-3: CIRCLE OR LOOP-BACK TEST

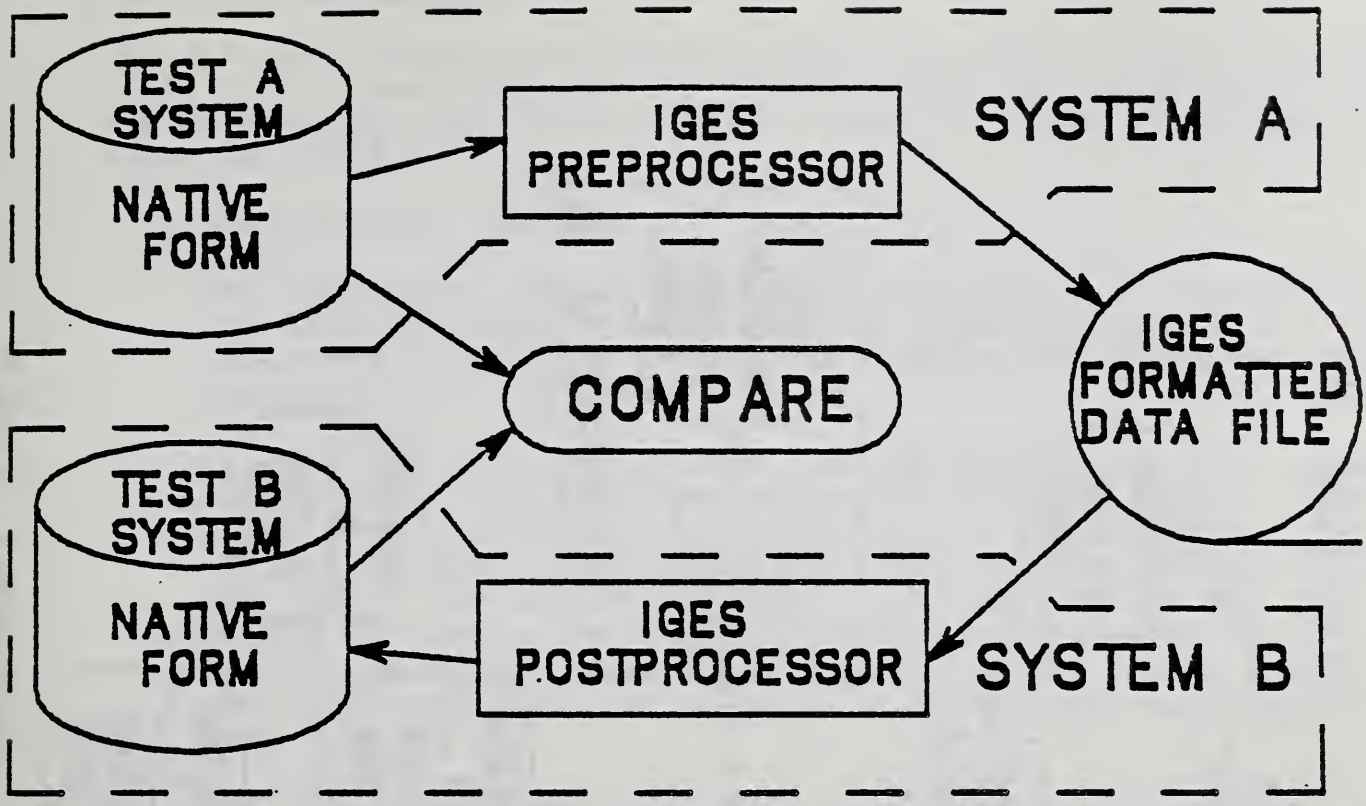


FIG. 6-4: SIMPLE INTERSYSTEM TEST

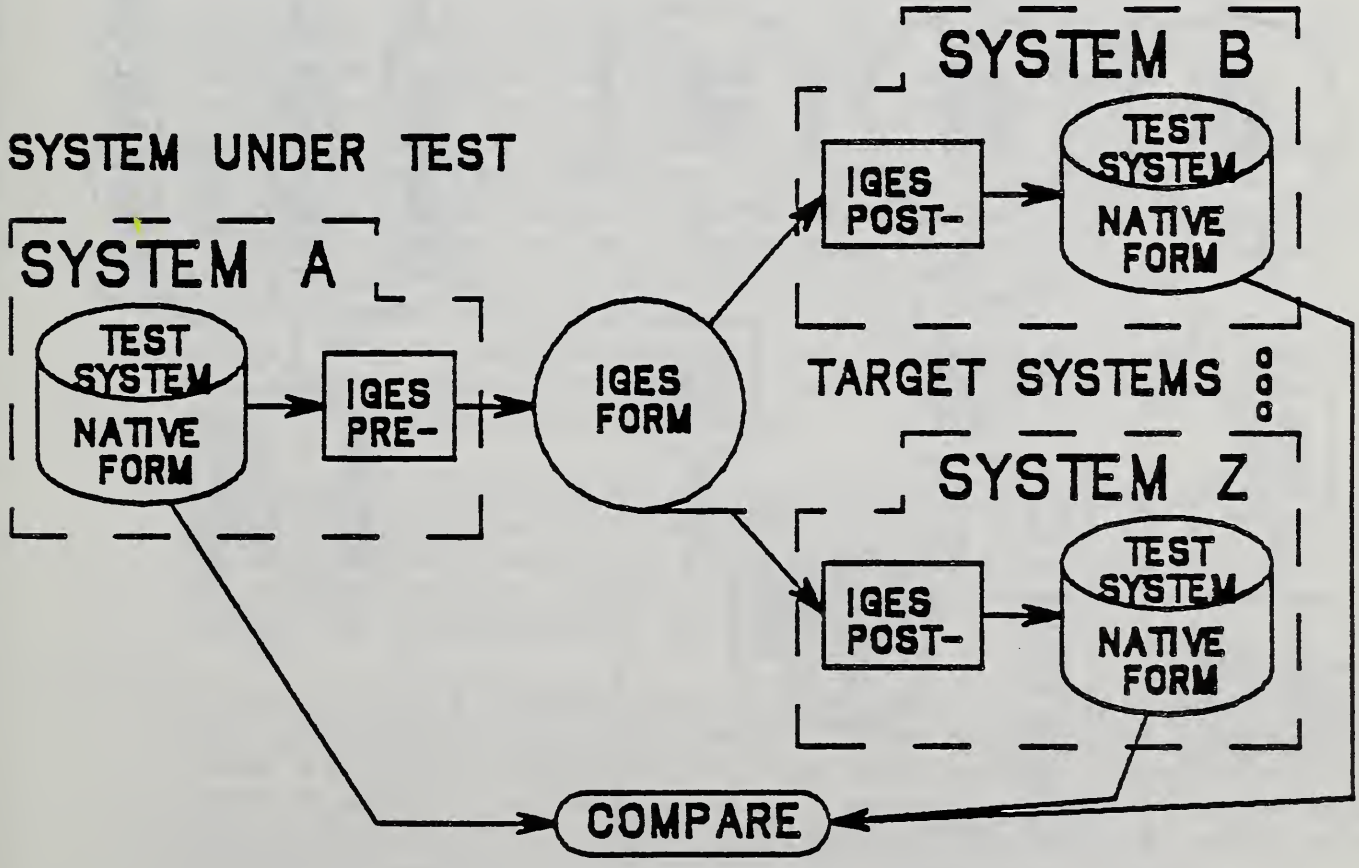


FIG. 6-5: JURY SYSTEM TEST

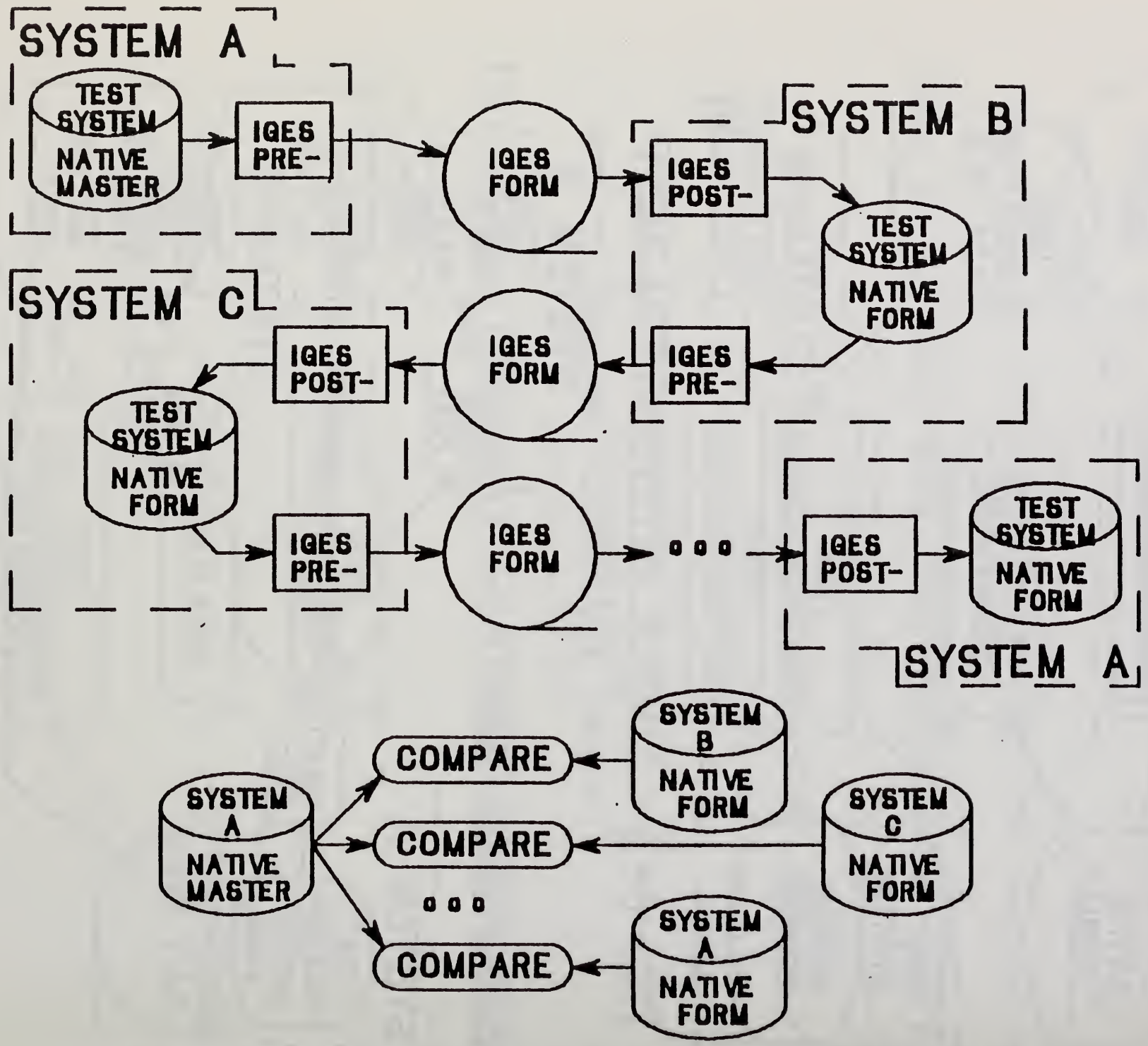


FIG. 6-6: DAISY CHAIN SYSTEM TEST

7.0 VERIFICATION TESTING

The following paragraphs describe how the verification tests are to be run. Some variation on the test methods explained below may be used for particular tests. Specific test procedures are given in Section 9.

7.1 Preprocessor Testing

The procedure used in testing preprocessors at the physical and logical levels (Table 6-1) is shown in Figure 7-1. The following paragraphs provide more detail about the various components of the test.

A script is provided in the test documentation describing how to model the Standard Reference Model on the CAD system under test. This script is functional in nature and identifies the kinds of entities and groupings needed for the test. The script is used by an experienced CAD user to create the native form Standard Reference Model. Once built, the model is verified to ensure that the proper entitites and data are present in the CAD system's data base.

The native form Standard Reference Model is then translated into the IGES form Test Result Model by the preprocessor under test. The preprocessor error log is used to aid in identifying problems with the translation.

Following the translation, the Test Result Model and the IGES form Standard Reference Model are compared for equivalence. The comparison must consider not the record-by-record equivalence of the IGES files, but must fully reduce the IGES data in order to establish the functional equivalence of the models. In this sense, not only must geometry be transformed completely into model space, but entity replications must be performed (e.g., groupings of entitites must be maintained). Display of the models may be used depending on the particular test being run. The comparison also produces exception messages describing any anomolies found.

During the comparision of the two models, some computer-based analysis may be required. These routines will be used for such comparisons as determining whether two space curves are equivalent by comparing their values at a prescribed number of intermediate points.

As part of the evaluation of the results, various verification procedures are followed. These procedures cover such items as syntax checking of the Test Result Model and collecting model statistics (e.g., entity counts, levels used, fonts used, etc.). The verification procedures produce exception messages describing any anomolies found.

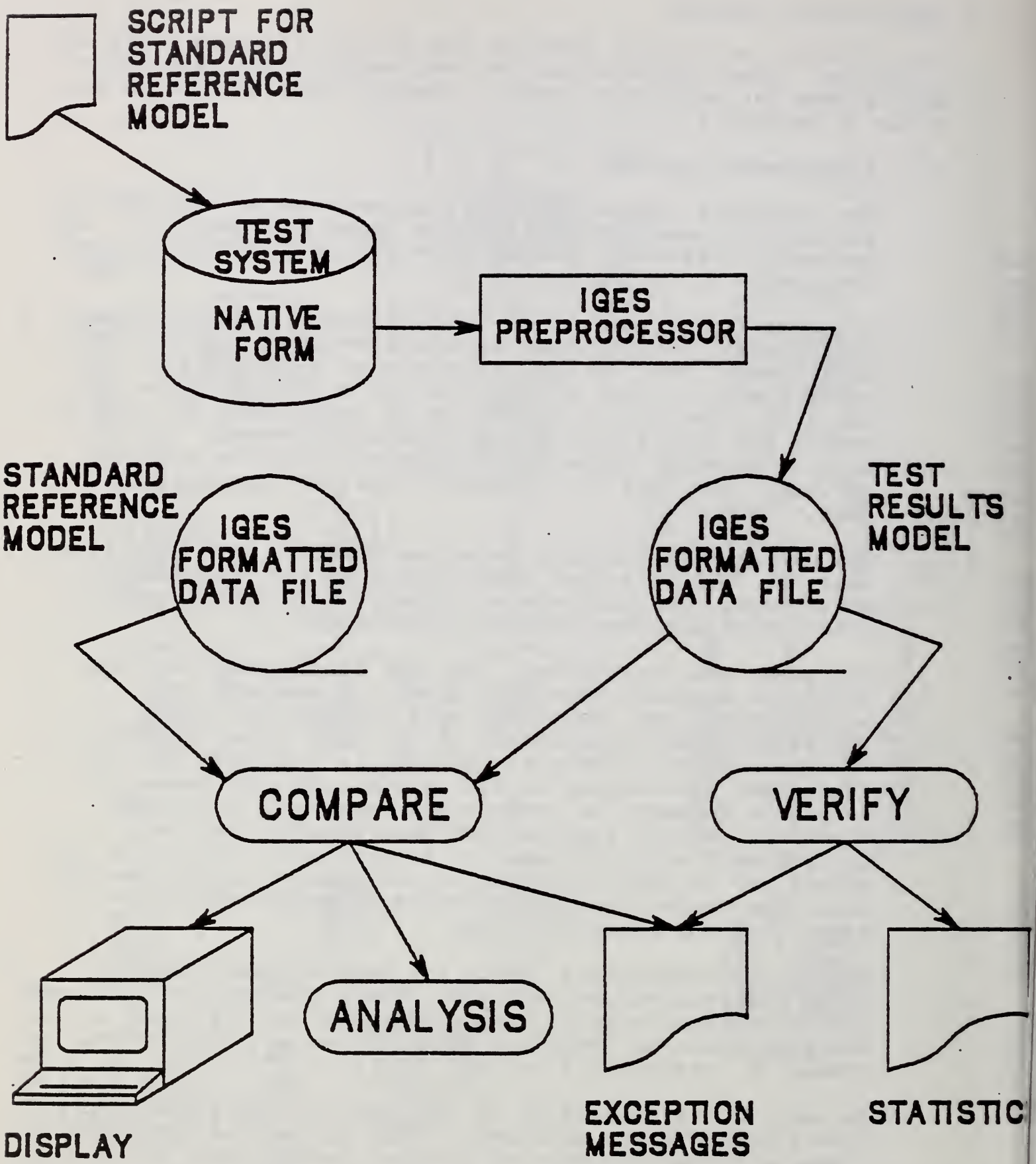


FIG. 7-1: TEST SCHEME ADOPTED FOR IGES PREPROCESSOR TESTING

7.2 Postprocessor Testing

The procedure used in testing postprocessors at the physical and logical levels (Table 6-1) is shown in Figure 7-2. The following paragraphs provide more detail about the various components of the test.

The IGES form Standard Reference Model provided in the test documentation is translated into the native form Test Result Model by the postprocessor under test. The postprocessor error log is used to aid in identifying problems with the translation.

The Test Result Model is verified for geometry, functionality, and operational completeness. A verification script is provided in the test documentation which describes the procedures to be used in evaluating the Test Results Model. This verification script identifies steps to be taken by the tester in checking that data and relationships have been preserved in the translation. The script is used in conjunction with other verification procedures and software. These procedures cover such items as what inspection and querying should be done against the Test Result Model, what further processing should be done against the Test Result Model, what further processing should be attempted on the Test Result Model on the system under test, and collection of model statistics (e.g., entity counts, levels used, fonts used, etc.). The verification procedures produce exception messages describing any anomolies found.

Another script is provided in the test documentation in some cases to describe how to model the Standard Reference Model on the CAD system under test. This script is functional in nature and identifies the kinds of entities and groupings needed for the test. The script is used by an experienced CAD user to create the native form Standard Reference Model. Once built, the model is verified to ensure that the proper entitites and data are present in the CAD system's data base.

In addition to being subjected to the verification procedures, the Test Result Model may be compared with the native form Standard Reference Model for functional equivalence. For these purposes, functionality is defined as "the degree to which the processed part data is treated as though it was created on the system under test" {DRAT84a}. Display of the models may be used depending on the particular test being run. The comparison also produces exception messages describing any anomolies found.

During the comparison of the two models, some computer-based analysis may be required. These routines will be used for such comparisons as determining whether two space curves are equivalent by comparing their values at a prescribed number of intermediate points.

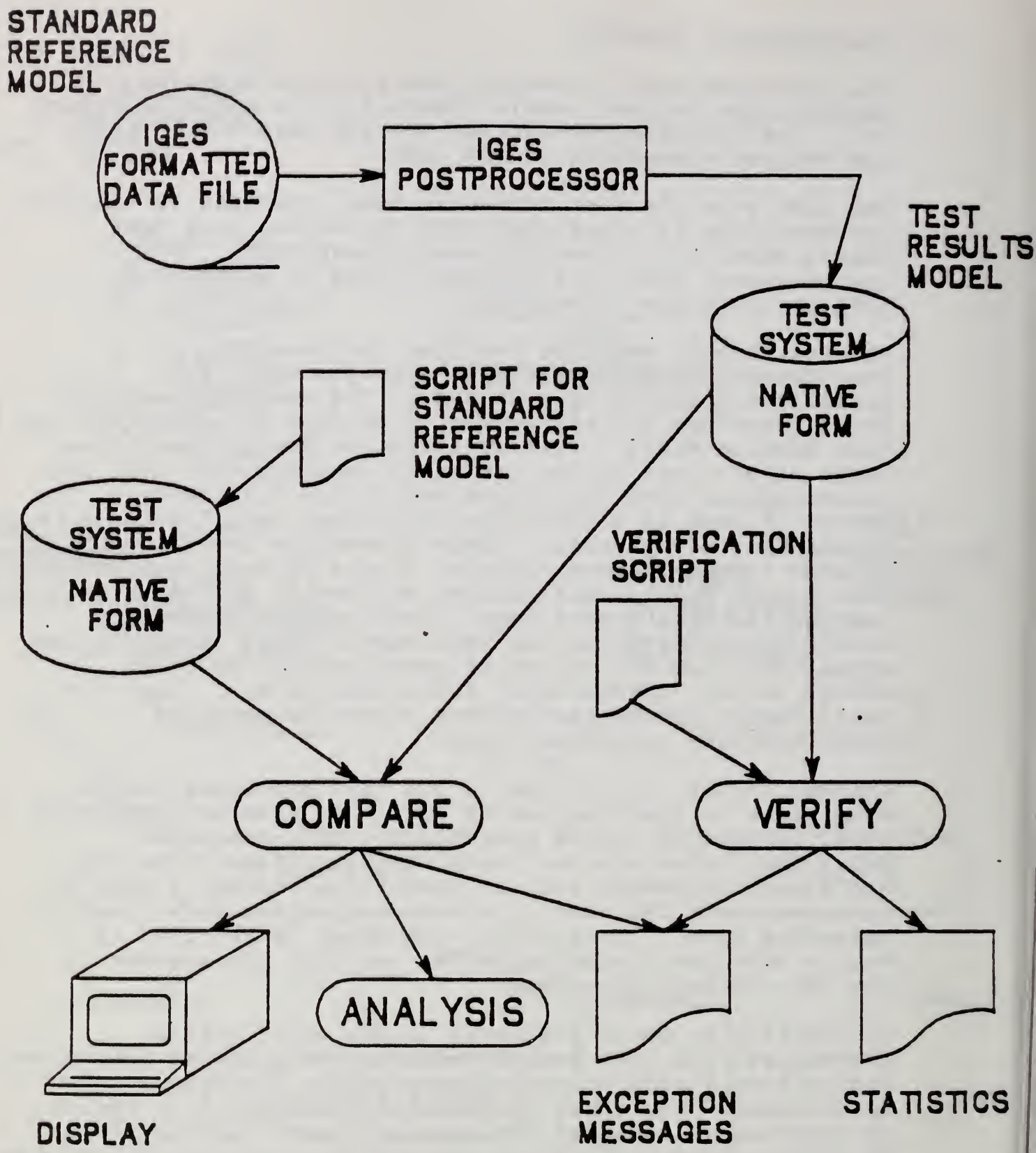


FIG. 7-2: TEST SCHEME ADOPTED FOR IGES POSTPROCESSOR TESTING

GRAPHICS INTERCHANGE

I.2 COMPUTER GRAPHICS STANDARDS

SPECIFIC TASKS

FY 86

1. Assess DoD needs for Computer Graphics (CG) interchange standards:

   a. Identify graphics interchange requirements in terms of CALS applications (e.g., engineering drawing repositories, spare parts procurement, technical publications).

   b. Assess the advantages and disadvantages of proposed and existing graphics standards for specific CALS applications. Recommend a set of computer graphic standards for DoD use. This should be done in conjunction with task I.1.2.a.

   c. Assess the compatibility of various graphics standards with proposed PDD standards (IGES, PDES, etc.).

   d. Assess current, intermediate and long term capabilities to apply CG standards to specific CALS applications. Identify and prioritize critical R&D issues.

   e. Identify priorities for key planning elements, including: testing for conformance to standards, testing for required performance, use of microcomputer graphics, and interface with database management systems.

   f. Develop a plan to expedite the development and implementation of CG interchange standards for CALS based on the above findings.

   Deliverables:

   -- Report to CALS Steering Group on tasks a-e (preliminary report three months after go-ahead, final report at six months)

   -- Plan for Computer Graphics area (outline three months after go-ahead, draft plan at six months, firm plan at eight months)

2. Accelerate CG standards development and validation efforts where needed to meet CALS schedule objectives:

   a. Accelerate and complete the ongoing NBS evaluation of the European validation suite. Use this as a baseline for recommending a DoD validation approach.

   * b. Develop preliminary functional specifications for validation suites for each of the CG standards recommended by task I.2.1.b (this might include CGM, GKS-3D, and/or PHIGS).

c. Evaluate the need for subsetting CG standards for CALS use. If needed, recommend an approach.

* d. Filter CALS graphic item requirements into the registration process.

Deliverables:

-- Quarterly status reports and a final report (eight months after go-ahead)

As DoD needs are determined, via the initial task, adjustments may have to be made to the remaining tasks. Tasks identified by an asterisk (*) appear to be low priority for FY 86. These tasks will be accomplished in FY 86 if possible. If not, they will be deferred to FY 87.

Tentative FY 87/88 Tasks

FY 87 and 88 tasks will be firmed up in the tactical plan delivered six months after FY 86 go-ahead. Tentative tasks include:

FY 87

1. Develop preliminary functional specifications for conversion of European validation suite to programming languages needed by CALS (e.g., Ada).

2. Complete enhancements to validation suite.

3. Complete conversion of validation routines to additional languages.

4. Produce report on microcomputer based graphics.

5. Begin development of CGM validation suite.

6. Begin development of GKS-3D validation suite.

7. Begin development of PHIGS validation suite.

FY 88

8. Complete CGM validation routines.

9. Continue PHIGS validation routine development.

10. Complete GKS-3D validation routines.

11. Produce report on benchmarking methodology.

12. Demonstrate completed validation suites.

13. Demonstrate CGM transfer between CALS systems.

COMPUTER AIDED LOGISTICS SUPPORT (CALS)
FINAL REPORT
GRAPHICS STANDARDS

TABLE OF CONTENTS

COMPUTER AIDED LOGISTICS SUPPORT (CALS)
FINAL REPORT
GRAPHICS STANDARDS

1. INTRODUCTION

This section of the report is a final deliverable for Task I.2, Computer Graphics Standards for Logistics. It discusses two specific task areas: I.2.1) Assess DoD needs for Computer Graphics interchange standards; and I.2.2) Accelerate Computer Graphics standards development and validation efforts where needed to meet CALS schedule objectives.

The strategy in meeting the deliverables for CALS in FY86 was to involve knowledgeable NBS personnel as well as key people from the graphics standards community in the CALS effort. The involvement of key graphics standards personnel in the CALS work served a twofold purpose. First, it brought together the most knowledgeable graphics standards experts focusing on solutions for CALS. However, in order to recommend CALS solutions, these graphics experts had to become knowledgeable concerning detailed CALS requirements. Thus, the second and most important benefit of involving key national and international standards people in CALS is that these people now regularly <u>represent</u> CALS requirements during discussions at both national and international standards meetings. The end result is that CALS has had a significant impact in the standards community in determining priorities and future directions of graphics standards.

It is appropriate to draw attention to Appendix 1 of the report. It is recommended reading before continuing with the rest of the graphics section deliverables. A large part of NBS's job for CALS in this first year has been to educate DOD in the area of standards. As such, this appendix represents a substantial effort toward meeting this educational requirement. Appendix 1 is divided into three distinct sections, each of which can stand on its own as a separate reference. The first section describes the family of computer graphics standards in a manner that is understandable for anyone trying to gain some grasp of the what and the why of computer graphics standards. The material for this section has been obtained in part from Dr. Peter Bono's first CALS contract report [BON186], and should be read before proceeding on since it lays a proper foundation for understanding the discussions of and recommendations for the utilization of computer graphics standards in CALS contained in subsequent sections of the report. It has been deliberately separated out of the main report so that it could provide a short, but complete reference to graphics standards, especially since so many DOD

people involved in CALS have expressed the need for a single source of information on graphics standards.

The second section of Appendix 1 provides the reader seeking understanding of NBS's mission and efforts with regard to computer graphics standardization; it also is a separate reference which can, as the first section of the Appendix, stand on its own.

Finally, the third section of Appendix 1 provides the reader an up-to-the-minute overview on where in the standards process the graphics standards are at the international (ISO), national (ANSI), and federal (FIPS) levels. It becomes important to know how standards important to the CALS effort are progressing in relation to the schedule demands imposed by the CALS program.

Section 2 details the work accomplished on subtask I.2.1.c to "assess the compatibility of graphics standards, both among themselves and with the product definition standards (IGES and PDES)." Appendices 2, 3, and 4 are integral parts of this effort, and formally specify how graphics standards relate to each other, how IGES entities could be rendered by graphics standards, and how PDES entities correspond to "elements" in the graphics standards. Recommendations for making these standards more compatible are discussed in Section 5.1. This work was also accomplished in part through Dr. Bono's first contract report [BON186].

Section 3 details the work accomplished on subtask I.2.2.a to "accelerate and complete the ongoing NBS evaluation of the European validation suite," while Section 5.2 discusses the second part of this subtask, namely "Use this as a baseline for recommending a DOD validation approach." This work was primarily accomplished through Dr. Steve Carson's CALS contract report [CARS86]. If the reader is interested in the nuts and bolts details of the validation testing, please read the contract report, which accompanies this report.

Subtask I.2.2.c was to "evaluate the need for subsetting Computer Graphics standards for CALS use." Based on our initial investigations of CALS requirements, it is premature to state whether or not subsetting will be necessary. Since CGM has been recommended as the standard of immediate use for CALS, closer inspection of CALS requirements is necessary before determining that, for example, only one CGM encoding should be used in all of CALS.

Section 4 discusses areas of the work under subtask I.2.1.a, namely to "identify graphics interchange requirements in terms of CALS applications." Appendix 5 contains NBS's preliminary thoughts on how particular CALS projects might utilize standards, and has been added here for completeness. Section 5.3 references

Appendix 6, which details (from the contract report [SPAR86]) a possible short and long term solution to the raster vs. vector problem for CALS. Section 5.4 then delineates the recommendations for future work on computer graphics standards for CALS. Appendix 7 is referenced here to show that NBS is aware of the implications for CALS of the work going on to incorporate CGM into the TOP industry standard, and that unnecessary deviation from these kinds of efforts should be avoided. Parts of this section of the report are attributable to Dr. Bono's second CALS contract report [BON286].

Finally, Section 5 collects all the recommendations that Sections 2, 3 and 4 generated, which serve as the end-products of the contract reports, and represent the contractors' "wish lists" for all the things that should be done in the area of graphics standards for CALS. NBS does not recommend that DOD fund all activities identified here. However, contractor-recommended lists are provided for completeness. The proposed FY87 Statement of Work contains the prioritized list of NBS tasks in the area of computer graphics standards. This, in effect, is the draft plan called for in subtask I.2.1.f, "develop a plan to expedite the development and implementation of Computer Graphics interchange standards for CALS."

In addition, Section 5.5 provides a summary of the prioritization for subtask I.2.1.e, to "identify priorities for key planning elements, including testing for conformance to standards, testing for required performance, use of microcomputer graphics, and interface with database management systems." NBS/ICST has incorporated priorities for graphics standards efforts in the proposed Statement of Work.

2. COMPATIBILITY OF GRAPHICS AND PRODUCT DATA STANDARDS

2.1 CRITERIA FOR FUNCTIONAL COMPARISONS

As has been documented, there are a number of graphics-based exchange formats, which have been or are being standardized. Comparing them is difficult, because they have been designed to meet a variety of objectives. Nevertheless, there are some criteria that are important across all such picture exchange standards and are useful in assessing which standard to use, if the application does not fall into the normal scope of just one of the exchange standards.

2.1.1 Representational Power

One of the simplest measures of the utility of a standard is its representational power. This means how rich is the standard in its primitive elements for representing pictures. For example, the Graphical Kernel System Metafile (GKSM) and Programmers Hierarchical Interactive Graphics System (PHIGS) have only 5 basic output primitives, while the Computer Graphics Metafile (CGM) has 18 different output forms. The product data base standards have many more basic entities, because they are more application oriented. Many specialty items for certain application areas--mechanical design, electrical design, etc.--are available for use.

Also to be considered is whether the standard can directly represent views of three dimensional objects or whether the application must first do the 3D-to-2D projection and viewing before describing the object in terms of its 2D projection.

2.1.2 Expressive Power

Another measure of the power of an exchange standard is the expressiveness of its semantics. Support for structures and segments allow the representation of more complex entities and relationships. The presence of a "macro" facility will also facilitate representing these relationships. Finally, if the semantics permit alternate representations, which depend upon the settings of certain parameters (e.g., ABSOLUTE or SCALED linewidths), more kinds of pictures can be represented and portrayed in a device-independent manner on a wider variety of workstations.

2.1.3 Extensibility

The process of developing a formal standard is a lengthy one: 4 to 7 years depending upon complexity. And the review cycle for ANSI standards is 5 years. This means that a standard must provide some mechanism for private extensions to meet needs that were not considered during the development of the standard. Without such extensibility, otherwise useful and valuable standards may be ignored or neglected by applications which need, say, just 10% more functionality than is provided by the standard.

When mechanisms for registering these application-specific extensions exist, the extensions achieve a semi-formal status and are more likely to foster interchange. The graphics standards--GKS, PHIGS, CGM, and Computer Graphics Interface (CGI)--are all beneficiaries of a registration mechanism being formalized by the International Standards Organization (ISO). The NBS/ICST is the international Registration Authority for the Register of Graphical Items, and thus can represent CALS needs for extensions to the graphics standards.

2.2 CRITERIA FOR SYNTACTICAL COMPARISONS

Semantics determines what kind of information can be represented in an exchange file; syntax determines how that information is expressed in concrete terms. The same semantics can be represented by different syntaxes; depending upon the purpose of the exchange file, different criteria are important.

2.2.1 Speed of Processing

If the graphics information is stored in a format that is close to the internal number representation of the host computer, interpreting that file will take a lot less time on similar machines. Of course, when transported to dissimilar computer environments these so-called Binary Encoding formats may be very slow to process, because of the need to convert all values from one internal binary format to another.

Another factor that affects the speed of processing is how complex are the rules used to encode and decode the data. Simple rules lead to fast processing. Unfortunately, they are also often associated with large files: only binary file formats are both fast and reasonably compact.

2.2.2 Compactness

File size is often a very critical factor, especially on PCs and in networks, where each workstation may not have a large amount of secondary storage. File size is also crucial where the files are sent via telephone networks, where character transmission rates of 1200, 300, and even 110 baud are most commonly encountered.

Techniques developed by the CCITT and the Character Coding committees of ISO (TC97/SC2) and ANSI (X3L2) are often used as the basis of the most compact picture and text representation formats. Typically based on the ISO 646 (ASCII) 7-bit and 8-bit character codes, these techniques are present in the CGM Part 2 and many proprietary coding techniques used by a wide variety of graphics peripheral devices.

2.2.3 Human Understanding

Sometimes it is more important that an exchange file be readable by humans. The CGM Part 4, the so-called Clear Text Encoding, and one version of IGES were specifically designed with human comprehensibility in mind. Generally speaking, these formats restrict themselves to the printing character set of ASCII and use mnemonic abbreviations to represent the basic elements of the file.

Standard text editors can be used to create such exchange files. This becomes useful for testing purposes. And being able to read an exchange file makes debugging more convenient.

Because English text contains a lot of redundancy, these files are not very compact nor are they necessarily fast to process, because the words must be parsed and converted to numbers via table-lookup before being processed.

2.2.4 Consistency with Existing Standards

Where new exchange standards are consistent with existing standards, production efficiencies can be obtained by manufacturers by designing algorithms and building firmware and silicon to process these standardized formats. The common encodings of the CGM and the CGI are expected (within the next three to five years) to bring wide performance and manufacturing efficiencies to graphics terminal and peripheral device manufacturers.

These efficiencies can accrue to the users of the standards because of the semantic and syntactic match between the applications using the standards and the hardware devices

implementing and supporting the standards. For example, the new Office Document Architecture/Office Document Interchange Format (ODA/ODIF) standard (ISO DIS 8613) will include a Geometric Graphics Content Architecture, based on the CGM semantics and syntax. Consequently, applications involving mixed text and graphics--like technical documentation and diagrams for logistical support--should benefit from devices and systems supporting the CGM and CGI encoding formats.

2.3 FEATURE COMPARISONS AMONG GRAPHICS STANDARDS

Appendix 2 details the specific graphics-related functions and how they relate among the graphics standards, in particular for GKS, GKSM, CGI, CGM, PHIGS and the PHIGS Archive File (ARCH). It provides a feature-by-feature look the functionality of the graphics standards, giving the reader a feel for which functions exist in which standards, as well as an easy reference for identifying areas that are not well-related between them.

No attempt has been made as yet to determine a priority for how to approach the many areas identified in Appendix 2 that need to be worked on to make these standards more compatible. At present CALS is only concerned with the CGM, so it perhaps is still premature to worry about the compatibility of all the graphics standards in general. Thus, most inconsistencies that have been identified will be items for the long term.

2.4 TRANSLATION OF IGES ENTITIES INTO GRAPHICS ELEMENTS

The IGES file format treats a product definition as a file of entities, each entity being represented in an application-independent format. The entity representations available in IGES include forms common to current CAD/CAM systems and forms that support the systems technologies currently emerging.

Entities are categorized as geometric and non-geometric. Geometric entities represent the definition of the physical shape of the product. Non-geometric entities typically serve to enrich the product definition model by providing a viewing perspective in which a planar drawing may be composed and by providing annotation and dimensioning appropriate to the drawing. Non-geometric entities further serve to provide specific attributes or characteristics for individual entities or groups of entities and to provide definitions and instances for groupings of entities.

The entity set of IGES allows the description of wire-frame models. Issues of compatibility with graphics standards must, therefore, involve whether the graphics standards have a rich

enough vocabulary of primitives and attributes to produce views of these wire-frame models in an efficient and accurate manner. Appendix 3 describes each of the IGES entities in turn, noting how well these entities could be rendered by a system based upon the graphics standards. It does not attempt to make any decisions on which particular IGES entities should be added (in some fashion) to the graphics standards. This is a subject for future efforts.

2.5 CORRESPONDENCE BETWEEN PDES ENTITIES AND GRAPHICS ELEMENTS

Appendix 4 details the specific areas of the PDES specification which discuss the PDES viewing pipeline, pictures, text model, color model, line attributes and surface attributes, and how they correspond to graphics elements within the graphics standards. The PDES presentation entities were deliberately borrowed from the graphics standards. With a few exceptions, it would be trivially easy to use an application program written using one of the graphics programming standards--PHIGS, GKS, or CGI--to interpret the PDES presentation entities and render the image correctly on a graphical output device.

2.6 DISCUSSION OF GLOBAL CONSIDERATIONS

It must be emphasized that IGES and PDES are not directly comparable with the graphics standards--GKS, PHIGS, and CGI/CGM. The product definition standards have quite different objectives than the graphics standards. Nevertheless, it is useful to examine the standards for compatibility one to another, because one would expect to both read and write IGES/PDES data bases from application programs written using GKS, CGI, and especially PHIGS. Furthermore, one would expect to extract drawings from IGES/PDES data bases and represent them as CGM files, before incorporating them in manuals, reports, and logistical support systems.

2.6.1 Dimensionality and Coordinate Systems

IGES and PDES deal with 3D wireframe objects; so do PHIGS and GKS-3D. At present, neither CGI nor CGM accommodate 3D objects. In all the 3D standards the coordinate systems are right-handed.

2.6.2 Viewing Models

The viewing models of PHIGS, GKS-3D, and PDES are much richer than that for IGES, since only one type of projection--an orthographic parallel projection--appears to be supported by IGES.

In recognition of this fact, PDES has deliberately borrowed most of the PHIGS viewing model (which is also shared with GKS-3D). However, in its current draft, PDES restricts the transformation matrix to 4x3, rather than the fully general 4x4.

PDES includes the workstation transformation--a device dependent transformation--in the definition of a view, unlike PHIGS, which does not allow the workstation transformation to be changed during structure traversal.

The 2D standards--GKS, CGI, and CGM--can be looked at as trivially embedded in the more complex viewing model of the 3D standards.

2.6.3 Text Model

The IGES text model superficially resembles the text models of the other standards, but there are many differences. For example, in IGES--as in GKS and PHIGS, but unlike CGI and CGM--a single font number or index applies to the whole note and combines the separate concepts of type face and character set. In IGES, only 7-bit character codes are supported.

Unlike any of the other standards, IGES uses a single index, the form number, to represent many independent aspects of text: single or multiple strings, justification of the strings within a text rectangle, the presence of subscripts, superscripts, fractions, and font changes. Special cases have to be added continually to handle new layouts.

PDES has taken its text model from CGI/CGM. The GKS and PHIGS text models are subsets of the CGI model.

2.6.4 Relationships Among Entities

IGES/PDES supports very complex relationships among product entities. Not only can entities refer to one another, but a macro facility is provided to allow the creation of application-specific entities and relationships. Furthermore, many entities are specifically present to represent special relationships (e.g., OFFSET CURVE, OFFSET SURFACE, and CONNECT POINT). At a higher degree of interrelationship, there are several entities to support finite element modelling and the representation of drawings. Finally, there are numerous structure entities, like ASSOCIATIVITY DEFINITION and ASSOCIATIVITY INSTANCE, that permit the representation of arbitrarily complex networks of blocks of simpler IGES entities, including wireframe and surface definitions, annotation, and

properties. SUBFIGUREs are available to allow instances of objects--defined once--to be included in several places.

PHIGS includes a hierarchical geometric structuring facility for modelling graphical objects. Although nowhere as rich as IGES/PDES, PHIGS structures are more flexible and more powerful than the segmentation features of GKS and CGI.

At present, CGM has no capability to express relationships among the graphical primitives that comprise a metafile picture. Thus, modifications of groups of entities in graphical pictures are more difficult to accomplish. An addition to the CGM may incorporate the capability to provide these kinds of relationships. Since this capability would be desirable to have in CALS, ICST will work within the standards community toward this addition.

2.6.5 Non-graphical Elements

Non-graphical elements are useful for extending the standard and for storing information closely related to a standard product, but not properly a part of the standard representation.

All the graphics standards contain an APPLICATION DATA element to permit the communication and storage of non-graphical information. Furthermore, they include a MESSAGE element to allow for the transmission of information to an operator of the graphics equipment.

In IGES/PDES, the PROPERTY entity plays a similar role. Many engineering-specific properties are defined in the IGES specification. A few of them do correspond to graphics elements and would be used in rendering a picture; e.g., drawing size, drawing units, and region restriction.

2.6.6 Extensibility

In IGES/PDES, the MACRO capability allows the standard to be extended far beyond the common entity subset, utilizing a formal mechanism which is a part of the IGES specification. Implementing an interpreter for the MACRO facility is a major investment and may not be available on many systems. In the graphics standards, two elements, ESCAPE and GDP (Generalized Drawing Primitive), are available for implementors to extend the standard beyond the original specification. GDP is used to provide new graphics primitives and attributes, while ESCAPE is reserved for elements that do not directly create output on the view surface.

In IGES/PDES, there is no formal mechanism to exchange information about useful extensions. However, for the graphics

standards, a set of formal procedures have been written (subject to ISO approval) to establish and maintain a Register of Graphical Items. The register will record extensions to the graphics standards of the following sorts: new GDPs and ESCAPES; new line types, marker symbols, and hatch styles; associations between font indices and the appearance of characters making up the font; and errors. As previously stated, NBS/ICST is the Registration Authority for these standards.

2.6.7 Conformance Rules

All the standards that relate to product or picture exchange--IGES/PDES and CGM--have very weak conformance requirements. The specifications describe the file structure and syntax, but no requirements are placed on the generators or interpreters of IGES, PDES, and CGM files. The result is that conformance to the standard is trivial to verify, but doesn't assure you that two people receiving the same file will eventually see the same picture. In the case of CGM, CALS needs assurance that CGM implementations fully transfer the graphical picture from one device to another. ICST will work toward enhancing the conformance requirements of CGM.

On the other hand, the programming standards--GKS, PHIGS, and CGI--have rather strict conformance rules built into the standards. The rules serve two principal purposes: to assure that all implementations of the standard provide some minumum functionality and to guarantee that the behavior of all implementations is similar with respect to graphical input and output. Without the guarantees represented by the conformance clauses, application developers using a graphics standard would have no confidence in using any particular set of features from the standard. Instead, they would have to limit themselves to the most primitive elements to assure the widest support for their programs.

2.7 TRANSLATABILITY

2.7.1 Effect of Loss of Information

Much of the discussion above outlined how the content of IGES/PDES files could be expressed or interpreted by programs based on the X3H3 graphics standards. In many cases, it was noted in Appendix 3 how information is lost when going from a product definition file to a graphics representation. A few examples include:

-connectivity of points and lines

-directionality of lines

-offset relationships between curves and surfaces

-purpose of text: annotation vs. part of the model.

In many cases, it is advantageous that this selective information be lost! This is because the files become much smaller, are easier to interpret, and are much faster to process if a picture is the final object. This is precisely the case for Automated Technical Manual systems within CALS, which will be explained in greater detail in another section of this report.

If too much information would be lost, but most of the IGES/PDES product data is not needed, the CGM APPLICATION DATA element can be used to supplement the CGM elements to provide the needed relationships.

2.7.2 Consistency of Semantic Level

As one would expect, there is an unevenness of semantic consistency across the product definition and graphics standards, due to their differing objectives. However, when geometry is involved, it is desirable that both the product definition standards and the graphics standards be at the same semantic level.

The IGES standard is lacking in its viewing and text models. PDES is attempting to remedy this. The graphics standards lack richness in the area of output primitives: full conics, including hyperbolas and parabolas, should be available at a minimum. Extensions to support simple surface definitions are also lacking.

Among the graphics standards, the CGI lacks support for 3D, while the CGM lacks support for both 3D and any kind of structures including, but not necessarily limited to, GKS and CGI segments and PHIGS structures.

2.8 SUMMARY

Because there are a number of graphics-based exchange formats (each designed to meet different objectives), comparing them is difficult. However, this section has attempted to help the reader in the complex task of selecting the right exchange format. First, one must find a standard whose objectives are consistent with one's applications needs (functional comparisons). Then, one must hope that the selected standard has specified a syntax whose size and performance characteristics also meet one's needs (syntactical comparisons). Sometimes, it

may be necessary to select a standard based on its syntactic and representational qualities, rather than on its intended scope. In these situations, one will use the extensibility facilities to create an exchange file that meets all one's needs.

Appendix 2 provides the specific comparisons of features among all the graphics standards. Although acting on all these points of comparison is premature as far as CALS is concerned, it is important to lay a proper foundation that accurately specifies the areas of compatibility (and incompatibility) among graphics standards.

In a like manner, Appendices 3 and 4 provide comparisons of IGES and PDES entities with elements of the graphics standards, respectively. These comparisons are based on discussions of global considerations and transalatibility problems, and provide the necessary groundwork for future efforts aimed at making the graphics and product definition standards more compatible.

3. GRAPHICS STANDARDS VALIDATION EFFORTS

3.1 INTRODUCTION

This section provides a summary of results and a description of the approach taken in evaluating the European validation tests. Subsequent sections summarize the conclusions reached for each major category of test, namely data structure tests, error tests, and operator interface tests. For a more complete discussion of the validation tests run for the European Validation Suite, please read the attached CALS contract report [CARS86]. In Section 5.2, contractor recommendations are made regarding future work and the utility of the tests for DOD graphics validation purposes.

3.1.1 Statement of the Problem

The great diversity of graphics packages with different philosophies has inhibited the development of graphical applications software. Graphics standards, such as the Graphical Kernel System (GKS) and the Computer Graphics Metafile (CGM), have been developed to help eliminate this problem and to encourage portability between different environments. A validation procedure is necessary to insure that implementations of standards, such as GKS, are consistent with the standards. Without this consistency, the advantages of portability are lost.

Recognizing this, the European Community sponsored a series of workshops during 1981 and 1982 to develop a methodology for testing GKS implementations. The development of a suite of tests, based on the methodology developed at these workshops, was subsequently funded. The actual development work was done at the Technical University of Darmstadt in the Federal Republic of Germany, and the University of Leicester in England. The development work was supervised for the European Commission by the GMD (Gesellschaft fur Mathematik und Datenverarbeitung) in the Federal Republic of Germany.

The European test suite is supposed to subject a completed GKS implementation to a thorough test of its consistency with the GKS standards with hopes of discovering errors. The less errors a particular implementation generates, the greater degree of standardization it contains, yet the lack of errors generated does not guarantee correctness. As the test suites are developed further, the degree of confidence in the correctness of the implementation will increase. Basically, the GKS implementation is tested by calling GKS functions, sometimes in conjunction with input devices, which generate a response. The responses from these calls are then evaluated and corresponding error messages reported if the responses are not as those designated by the GKS

standard. The operator interface tests are evaluated a little differently since they require human visual evaluation of the graphical output from the device chosen for test.

The GKS validation test suite developed under the sponsorship of the European Community consists of three sets of tests. One set tests the data structures internal to GKS; a second set tests the errors that occur when executing GKS functions; and the third set tests the general functionality of GKS at the operator interface level. All of these tests are at the level of what would commonly be referred to as "system acceptance tests" in DOD terminology.

The data structure tests consist primarily of a dialog between the certification program and the GKS implementation. GKS routines are correctly called under valid states of GKS in order to determine the viability of the implementation. The responses of the GKS implementation are then written to a report file.

The error tests check the response of the GKS implementation to deliberately induce error situations. Specifically, the error messages returned by the implementation are compared with a list of correct responses, and reports of these comparisons are written to a report file.

The Operator Interface Tests consist of an "Operator Script" and output to the screen of a workstation. The Operator Script tells the operator what the screen should be displaying, and provides a form for noting the agreement or discrepancies between the scripted version and the actual content of the screen display.

The purpose of this effort was to ascertain the suitability of basing a DOD validation procedure for GKS implementations on the European GKS validation suite. If these tests are of sufficient quality, their use can save the substantial development effort needed to implement alternative tests. Section 5.2 details the recommendations made as a result of this study.

3.1.2 Approach to the Problem

To evaluate these tests, a three part strategy was devised. The first part of the strategy involved installing and executing the tests in a typical environment in the United States. The second part of the strategy involved analyzing the structure of the tests themselves to determine the quality of their construction and their modularity and ease of use in testing GKS implementations, especially on smaller computers or in embedded processors. The third part of the strategy involved a detailed investigation of the extent to which the routines test the requirements in the GKS specification.

3.1.3 Installation and Execution of Tests

The Validation Tests were developed in a European academic computing environment and have not seen extensive use within commercial production software development environments in the United States. By installing and executing the tests, it was determined that, in general, the quality of the installation instructions was disappointing, as was the ease of customization of the routines to a new environment, and the quality of programming practice used in their construction.

3.1.4 Summary of Results

The test suites were successfully installed on a Digital Equipment Corporation VAX 8600 computer running the VMS operating system. The graphics capabilities available on this computer include the ISSCO and the Digital Equipment Corporation (DEC) implementations of GKS. Graphics output devices included a Tektronix 4128 terminal and DEC VT240 terminals. A moderate amount of difficulty was encountered in installing the tests due to the poor quality of the documentation furnished with the test suite. For example, instructions detailing how to customize certain subroutines for a particular GKS implementation were contained in the written documentation accompanying the test suite and others in comments contained in the source code to the subroutines themselves. For several routines, both sets of instruction had to be followed.

The error tests and data structure tests were both completely executed using the ISSCO GKS implementation and with the VT240 terminal. A large number of error messages were generated as a result of the execution of these tests. For the data structures tests, some of the errors were programming errors in the GKS test programs and some were errors in the ISSCO GKS implementation which were discovered by the tests. Problems encountered executing the error handling tests were due mostly to programming errors in the GKS test routines. The operator interface tests were also completely executed with more favorable results. A moderate number of errors were encountered executing these tests with both the DEC VT240 terminals and the Tektronix 4128 terminal and most were due to errors in the ISSCO GKS implementation.

A determination of the structure and organization of the test programs was completed. During inspection of source code for the routines, numerous problems with programming practices used in their construction, the quality of their internal documentation and FORTRAN language coding errors were encountered. These problems are documented in the contract report [CARS86] in great detail. The overall impression is that the data structure and error tests are of very low quality. The operator interface test

source code is better, but still contains a number of errors and examples of poor programming practices.

The degree to which the validation tests determine conformance to GKS requirements was evaluated. This was done by taking a representative set of requirements, some of those associated with the POLYLINE output primitive of GKS, and determining if these requirements are tested in the validation routines. The more obvious requirements were usually well covered by tests at an appropriate level for an acceptance test procedure. Less obvious requirements were poorly tested or not tested at all. Some requirements were identified that could not be tested through the GKS subroutine call interface. Other forms of requirements verification, such as analysis or "internal unit-level" tests, must be used to verify these requirements.

3.2 REQUIREMENTS TRACEABILITY ANALYSIS

The quality of any validation test is determined by how well it tests requirements of the system being evaluated. This is especially difficult to determine in the GKS environment since the GKS specification itself is not as well organized as a traditional DOD System Requirement Specification. Nonetheless, evaluation of the test routines was undertaken by picking one area of GKS --the processing associated with producing the Polyline output primitive-- for extensive evaluation of requirements traceability. A partial list of GKS requirements relating to Polyline was developed and is contained in Appendix 2 of the contract report [CARS86]. Due to the extremely large number of Polyline requirements that were found, only a subset of them was able to be tested and evaluated.

Once these requirements were extracted from the GKS Specification, they were used to derive a minimal, testable set of requirements to validate that a GKS implementation conforms to the requirements for processing Polylines. The three sets of GKS tests were evaluated to determine how well they test each of the derived requirements. From this exercise, one can infer the degree of care used to construct GKS tests programs, the percentage of coverage of GKS requirements which they are likely to provide in an acceptance test situation, and the general quality of the tests.

The detailed results of the evaluation are presented in Appendix 2 of the contract report [CARS86]. In this Appendix, a representative set of requirements is listed, together with a description of how the requirement should be tested, an identification of one or more places in the validation suite where it is tested, and a determination of how well it is tested.

The degree of requirements coverage in the European validation suite is reasonable for an acceptance test situation, especially considering the lack of organization of the GKS standard and the difficulty of extracting testable requirements from it. All key features of GKS were tested in more than one way. However, meaningful requirements which were not adequately tested were easily uncovered, and requirements that could not be tested at the GKS language binding interface were found.

If additional confidence in the correctness of a GKS implementation is needed, then additional requirements verification must be performed. This could be done by analysis of the source code of the implementation or by demonstrating the correct performance of a set of "internal unit-level" tests designed to check the correct implementation of key features -- such as transformations and certain approximations-- that are not visible enough through the subroutine call interface to be adequately testable in a validation test suite.

3.3 CONCLUSIONS

3.3.1 Data Structure Tests

The internal documentation for the data structure test describes the overall effect of the test. The programming style and degree of commenting does not permit a quick reading of how the effect is achieved. This is a definite deficiency from a maintenance point of view.

The whole philosophy behind the data structure tests is flawed. Since no graphical output is produced during the tests, an implementation could pass by correctly implementing the update of certain internal data items in response to the invocation of GKS functions. Unfortunately, updating the data structures alone is not sufficient. An implementation must modify its future behavior --especially the graphical output it produces-- based on these values. This is not tested! The only valid way to test the GKS data structures is to interleave such tests with the production of graphical output. Demonstrating that a value in a state list can be set is of little value if the implementation makes improper use of that value.

Execution of the tests provides only a very minimal amount of information regarding the correctness of a GKS implementation. Many of the test routines perform functions other than those expected and many contain nearly identical code. The diagnostic messages provided are cryptic and nearly useless in locating and correcting defects in an implementation. The quality of their source code and documentation is extremely low.

3.3.2 Error Tests

The error handling test programs have a well defined structure of subroutine calls and the internal documentation is a bit better than the data structures tests. The simplicity of the error routines makes the code almost self-documenting without extensive commenting. Many errors are reported when they are executed. Most of these are due to errors in the test programs themselves rather than in the implementation being tested. Several fundamental design flaws in the tests prevent them from providing much useful information or from being properly evaluated. If time had been available to recode routines ESEXER and ECHEKR so that they worked correctly, a better evaluation of the error test could have been accomplished.

3.3.3 Operator Tests

The internal documentation in these tests is very thorough. A number of relatively minor coding errors were found in the test programs. Others probably exist that were not detected. The tests were much more effective than the data structure and error tests at uncovering problems in the GKS implementation being tested. These tests appear to be of some utility in validating GKS implementations.

3.3.4 Summary

Considerable effort will be required to bring the European validation suites up to an acceptable level for either DOD purposes or for use in "commercial" testing in the U.S. Effort will be required to properly document the internal structure of most of the tests, rewrite portions of them consistent with good programming practices, correct coding errors, and add additional tests to increase the degree of requirements validation. A detailed list of recommendations is given in Section 5.2.

4. ASSESSMENT OF DOD NEEDS FOR COMPUTER GRAPHICS STANDARDS

4.1 INTRODUCTION

Section 4.2 details work accomplished on subtask I.2.1.a, "identify graphics interchange requirements in terms of CALS applications (e.g., engineering drawing repositories, spare parts procurement, technical publications)." For a long time, this statement was interpreted to mean that for every CALS application identified and studied, NBS should make recommendations on their particular graphics standards interface requirements. However, this has proved to be too ambitious a task in the time provided, and has tended to fragment, rather than pull together, work on CALS overall standards needs. The services and DLA have identified more than 82 programs in their draft CALS implementation plans. It has been possible for NBS to visit or hear about (in sufficient detail) only a handful of CALS programs; and then these have been constrained to just two of the major CALS application areas, namely drawing repositories and technical publications. Therefore, NBS really does not have the total picture of CALS yet in any great detail. In addition, at the CALS Architecture meeting of October 7, 1986, it was decided to limit CALS projects to a few application areas for Phase I purposes.

However, quite a lot of NBS and contractor effort has gone into trying to satisfy this subtask based on providing recommendations for each CALS program. Efforts along this line include:

1) One of the contract reports [SDC 86], developed by Madeleine Sparks of the System Development Corporation, which attempts to describe each of the CALS programs (those for which information was provided to her) in terms of possible graphics standards usage (a copy of this report accompanies this deliverable); and

2) Appendix 5, which is a compilation of NBS efforts from all the standards areas in identifying those standards that could be applied to each of the CALS applications which NBS has studied. These represent are preliminary thoughts, and have not been refined. They are pointed out here for completeness.

Those efforts will not be reiterated here. Rather, in order to put this fragmented information into an overall perspective, this section describes the accomplishments on this subtask referred to above by documenting, using architectural diagrams, a framework for how the CALS program should use the computer graphics standards.

Although all the applications referred to in the following architectures for graphics data flow do not conform precisely to the same CALS programs identified as the 'representative' programs in the NBS Point Paper, CALS Representative Systems, most are the same. Any difference is due mainly to the fact that this section is confining itself to the flow of graphics data only. As a result, the architectures in this section do not address all of the CALS application areas.

Section 4.3 then tries to draw some conclusions from this discussion on the overall importance and impact of graphics standards on attaining the stated goals of the CALS program, while Sections 5.3 and 5.4 detail contractor recommendations that result from this discussion.

4.2 ARCHITECTURES FOR CALS

4.2.1 Overview

Most of the program elements needed for CALS either are themselves Computer-Aided Design (CAD) applications or share many characteristics with classical CAD applications. Before proposing specific architectures for the various applications important to CALS, it is useful to look at CAD in general.

4.2.1.1 General CAD Functions

When examining any system at a conceptual level, it is beneficial to think in terms of Inputs, Processes, and Outputs. For the typical CAD system, the inputs are drawn from the following information:

- specifications--WP files, OCR output, operator selections;
- drawings--output from raster scanners and automatic digitizers;
- external repositories--archived digitized drawings, CAD databases, component and symbol libraries;
- algorithms--finite element models, interference checkers, routing heuristics;
- practices--simulations, professional and legal standards, design rules.

Generally this information is read in and organized into two logically distinct, but often physically unified databases: the geometry model and the product database. Figure 4-1 illustrates this situation schematically.

Typical processing associated with a CAD system includes:

- Design and Drafting
- Revision
- Analysis
- Simulation
- Testing
- Scheduling
- Documentation

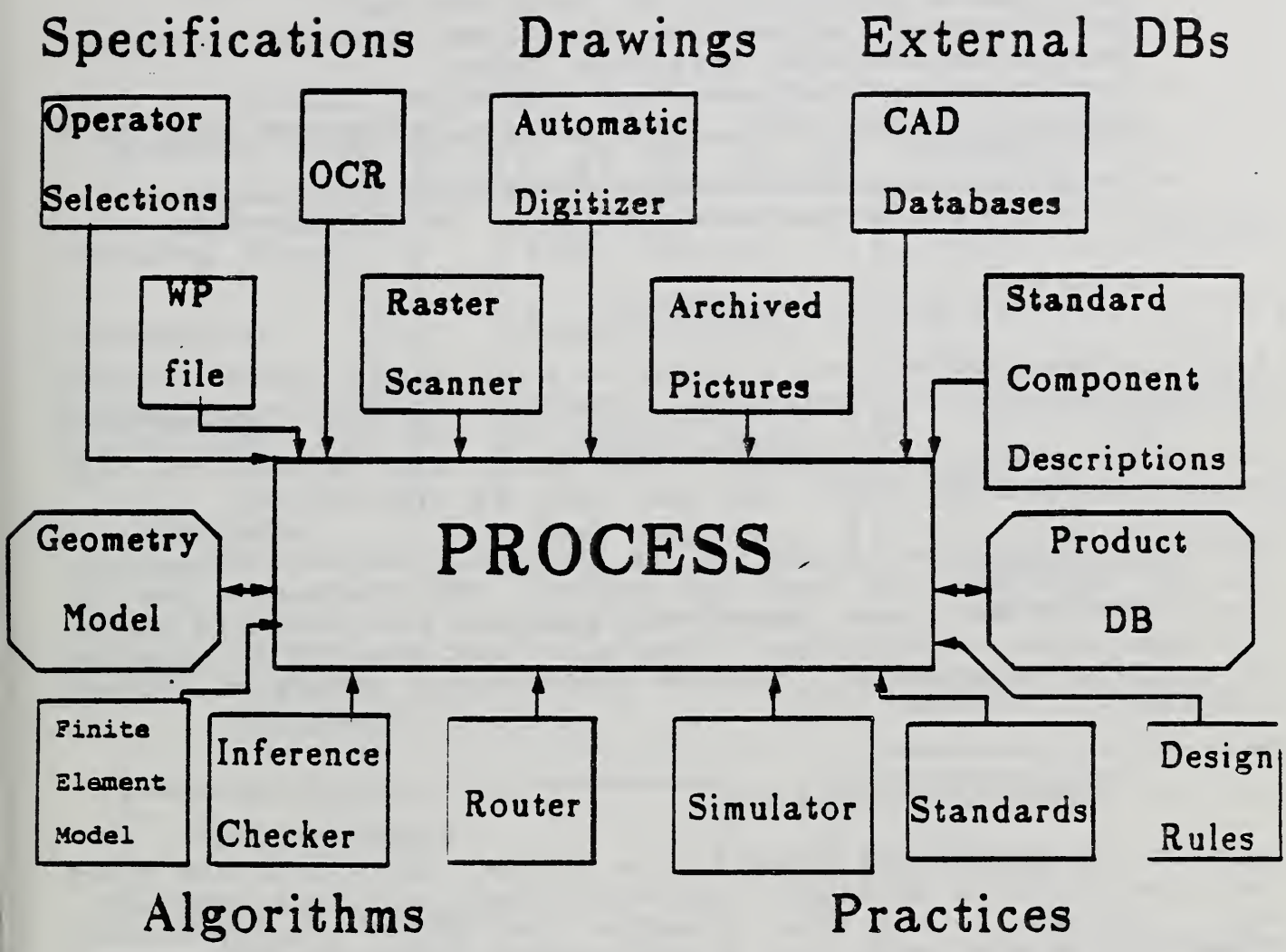


Figure 4-1. Sources of Information

CAD system outputs fall into a number of categories, illustrated in Figure 4-2 and described in the following:

- Drawings--detailed views, assemblies, and schematics;
- Analytical Results--FEM outputs, loads and weights, performance characteristics, test results;
- Manufacturing Instructions--N/C tapes, robotics commands, processing drawings, materials lists;
- Purchasing Instructions--Cost estimates, bills of material, parts lists, drawing lists, standards compliance requirements;
- Logistical Instructions--Documentation, parts lists, diagrams and schematics, cost data, re-ordering data.

4.2.1.2 General Picture Processing

A top-level architecture focussing on processing pictures, rather than on the specific components of certain applications is shown in Figure 4-3. The roles played by the API (application programer interface) standards (GKS, PHIGS, and CGI) and the data exchange standards (IGES, CGM, and SGML) is highlighted.

Note the importance of the CGI in providing device-independence for all devices--both input and output. New changes to the CGI specification have been tentatively accepted to accomodate raster and area input technologies. Note also that the CGM is expected to be able to store <u>all</u> picture descriptions generated through the CGI.

4.2.1.3 Kinds of Applications

Many of the inputs and outputs of a typical CAD system are mixed text and graphics documents, although some are purely graphical, e.g., drawings. All CALS elements that share characteristics with this general model can be addressed by studying the general model for opportunities and requirements for graphics standards.

CALS programs studied thus far fall into two major areas, each of which can be subdivided again into two areas. Automated data repositories store product and geometry models in a central database for shared use by a diverse set of people. The two principal application areas for CALS projects are Engineering Design and Procurement Support. Automated technical manual systems emphasize rapid access to current documentation, training, and maintenance information. Traditional printing and publishing applications fall into this category as do the more ambitious plans for paperless presentation systems and interactive delivery and maintenance systems. Table 4-1 lists some of the DOD projects that could contribute to the CALS objectives and the application areas that they fit into.

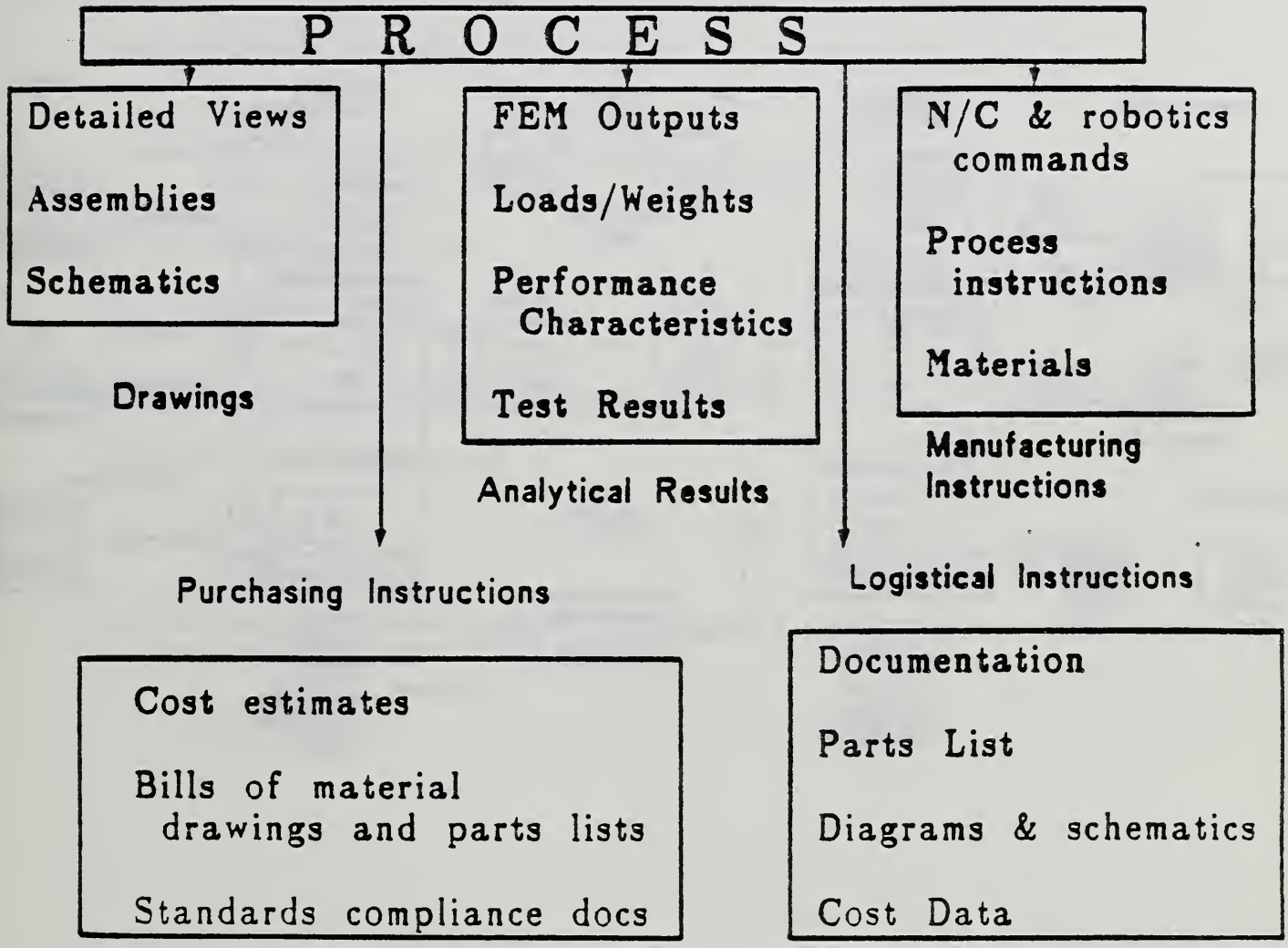


Figure 4-2. CAD Process Outputs

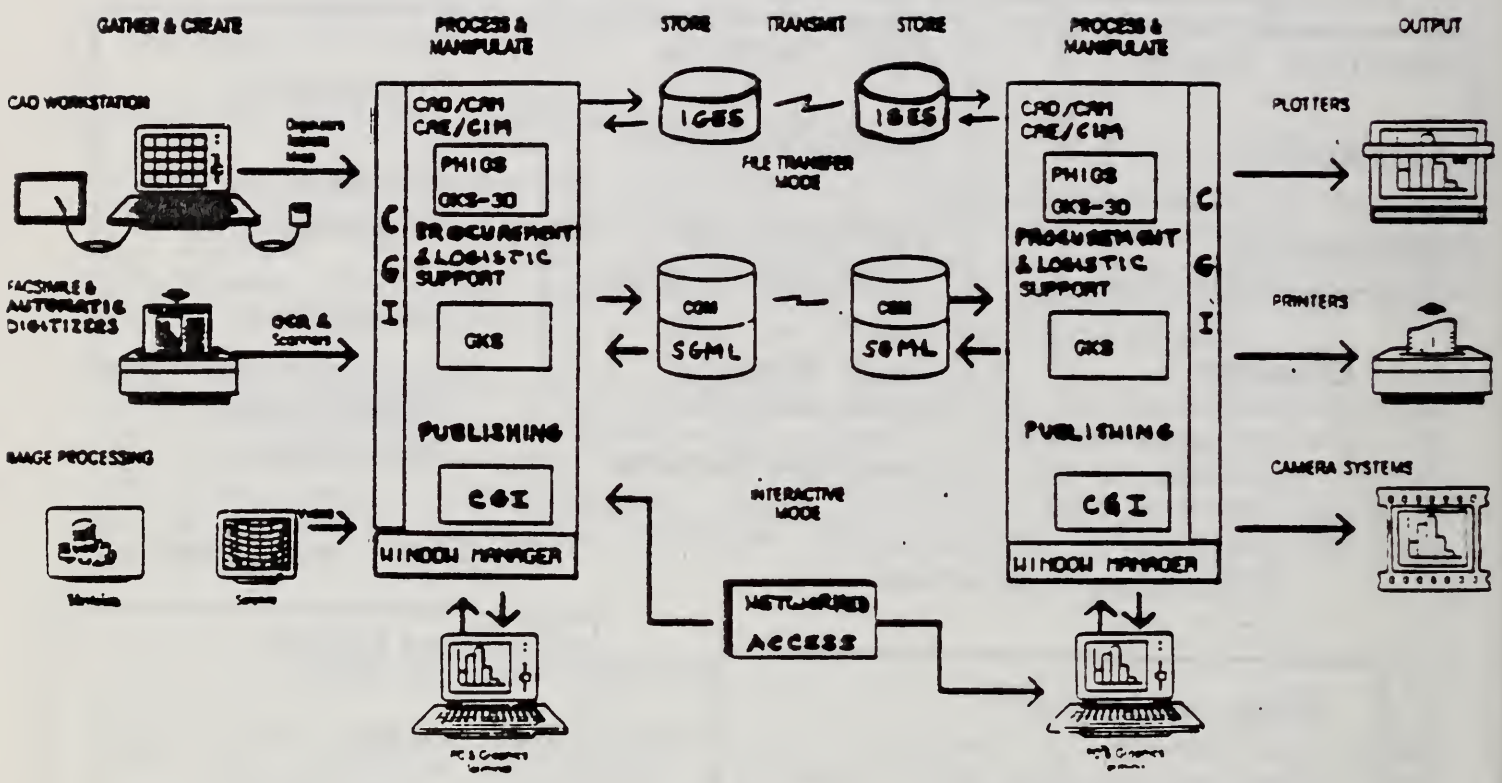


Figure 4-3. Picture Processing.

Table 4-1. CALS Application Areas.

AUTOMATED DATA REPOSITORIES

- ENGINEERING DESIGN
  - DSREDS
  - EDCARS
  - PDDI/GMAP
  - IDS
  - IISS
- PROCUREMENT SUPPORT
  - MASTER
  - UDB

AUTOMATED TECHNICAL MANUAL SYSTEMS

- PRINTING & PUBLISHING
  - APPS
  - ATOS
- PAPERLESS PRESENTATION / INTERACTIVE DELIVERY AND MAINTENANCE AIDS
  - EMPS
  - CMAS/IMIS

In the following sections, an architecture stressing the graphical aspects of each application area is presented. The specific inputs, processes, and outputs are listed. Finally, overall architectures for each of the two main application areas is portrayed. The architectures show how the various individual projects could exchange information using IGES, CGM, and pure image (raster) files.

4.2.2 Automated Data Repositories

4.2.2.1 Engineering Design and Drafting

Databases required to support CAD/CIM applications are very complex. Table 4-2 lists the principal tasks associated with managing an engineering database. Acquisition, storage, and retrieval of the drawing data and associated product information is a major undertaking--and costly too. All data must be kept in revisable form so that the database can track changes made in the design as the product moves from the early conceptual design stages through detail design, validation, and final manufacturing. Once in production, revisions will be needed; either to fix errors or to make improvements.

Figure 4-4 illustrates the principal interfaces surrounding an engineering database. Both IGES databases and CGM picture files should be used for importing and exporting information, but the database itself is likely to have a structure and content much richer than that currently supported by either IGES or CGM. The PDES (and ISO STEP) efforts are more likely to eventually be useful in standardizing this environment. The output of the drawing digitization process could be expressed as a CGM file. The application running on the CAD workstation used to create original drawings and modify existing drawings could be built upon GKS or PHIGS--themselves, perhaps layered, on top of CGI--to provide device-independence and code portability.

On very large projects, or where there are many thousands of drawings to be managed, a distributed database, using networks of design workstations to manipulate the information, could be established.

4.2.2.2 Procurement Support

Repositories for supporting procurement actions may be as large and as complex as engineering design databases, but they differ in several other regards.

First, as Figure 4-5 shows the principal output of the procurement process is a collection of "paper" (that is, the documents that form the RFP/RFQ package) rather than the product design. Second, as can be deduced from the processes outlined in Table 4-3, the database changes much more slowly over time than an engineering database. Third, the operator generally does not revise, to any great extent, the information in the database. Rather, the operator browses through the information, perhaps analyzing contractor- and government-supplied statistics about

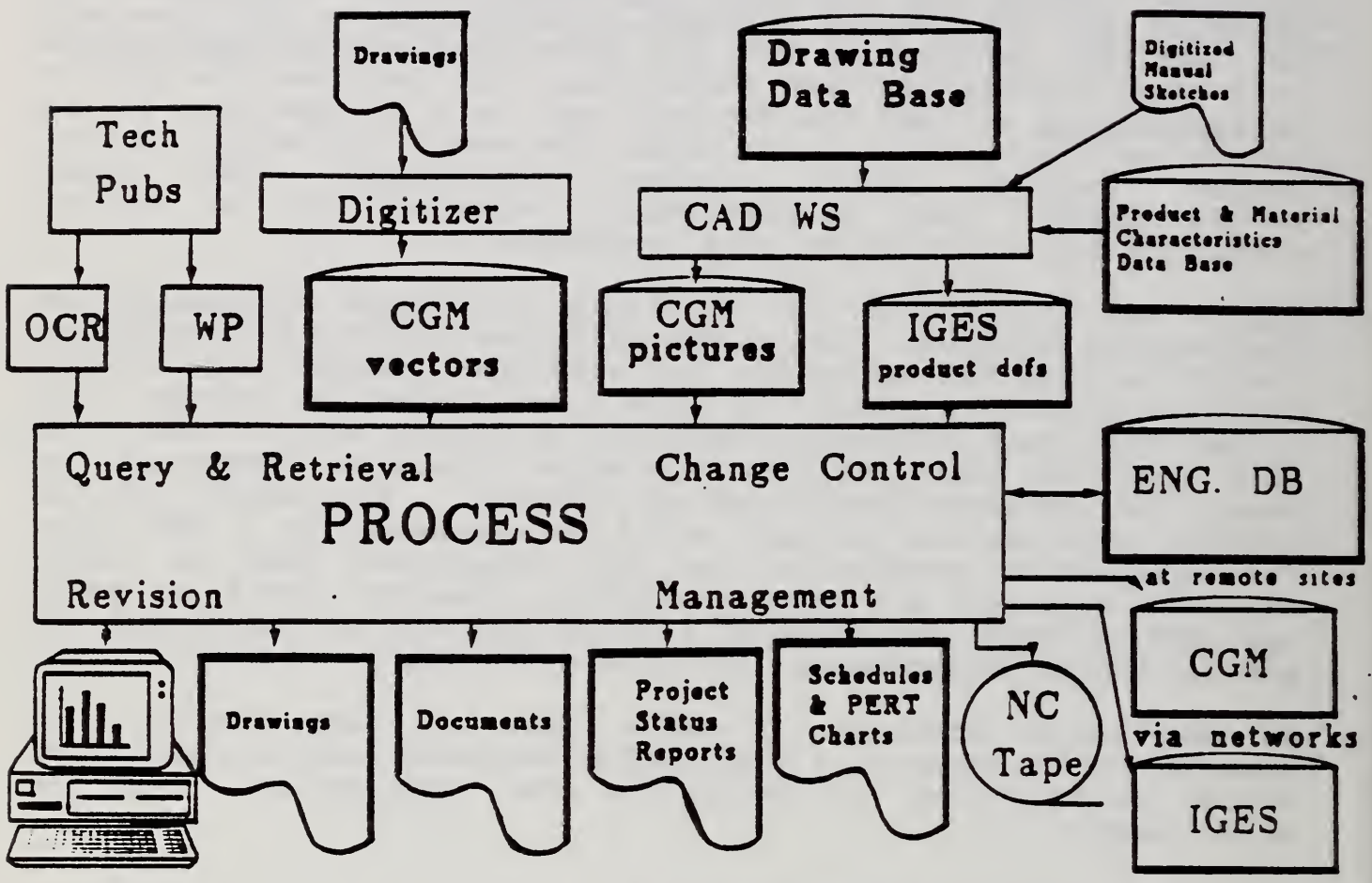


Figure 4-4. Architecture for Engineering Design Data

Table 4-2. Engineering Design Data Repository.

Input	Process	Output
Engineering Drawings	Scan & digitize	Engineering Drawings
Technical Drawings	Transmit to remote sites.	Technical Drawings
	Store in digital form.	
	Retrieve	Technical Documents
	Generate hardcopy.	
Geometry	Revise drawings (EDCARS)	NC & robotic program codes
Product & material characteristics	Allow for feedback loops	Reports, PERT charts, project management information
CAD/CAM WS for revision	(Digital) Product Definition Data	
	Process Management	

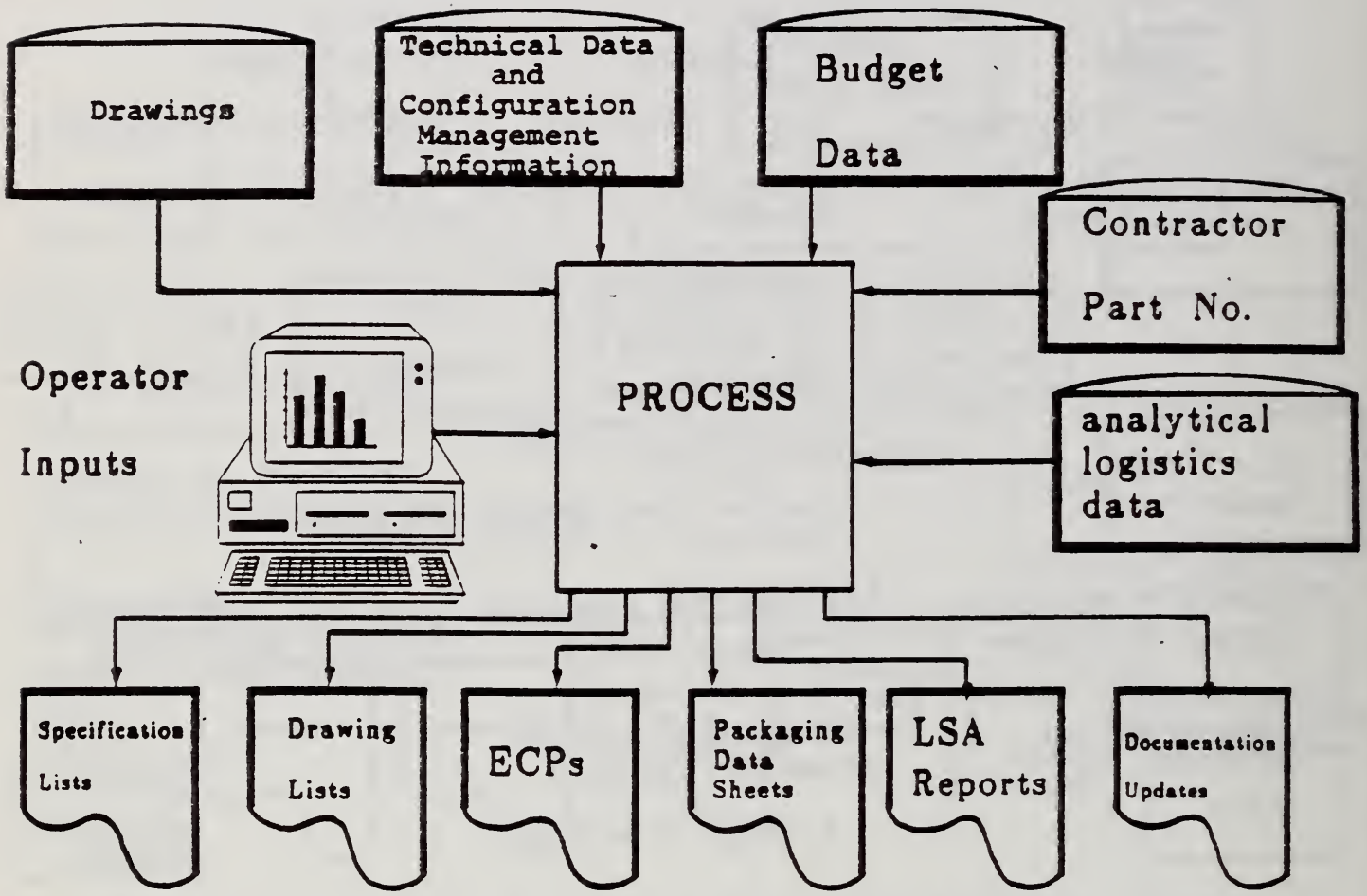


Figure 4-5. Architecture for Procurement Support

Table 4-3. Procurement Support.

Input	Process	Output
Technical Data/ Conf. Mgmt. System	Relate status of data to type of procurement.	Specification Lists Drawing lists
Budget Stratification System	Statistical Analysis	Packaging data sheets
Operator	Retrieval	Documentation update
Part number data from contractors		LSA reports
DS/REDS		Query outputs, including graphics
Logistics data (e.g., MTBF, MTTR, cost, availability, etc.)		Engineering change proposals

the performance of the parts stored in the database and ultimately making a series of selections that results in lists being generated. From time to time, documentation updates may be produced for incorporation into manuals and for wide dissemination to field activities and contractors.

Where picture storage and retrieval is required, the CGM and raster image files should be more than adequate for the needs of these kinds of applications. IGES, or any such product databases, are not needed and, indeed, would be quite inefficient for most applications. The related text standards--ODA/ODIF and SGML--could be used for storing and transmitting the documents themselves. Both standards could use the CGM to incorporate images into the running text of the documents.

In these applications, there is no particular need for API standards based implementations. However, where previewing of drawings is required, applications may want to build upon the API graphics standards to get the usual benefits of program portability, maintenance cost savings, and device-independence.

4.2.3 Automated Technical Manual Systems

4.2.3.1 Publishing

Figure 4-6 and Table 4-4 summarize the automated publishing environment. Typically, a manual, report, or other document will be prepared by the author on another, local word processing system. If diagrams are required, the pictures will also be prepared locally. When there is a need for this document to be published in volume and maintained as an official publication, the source text and pictures need to be transmitted to the automated publishing facility. Clearly, standards like SGML and ODA, along with CGM for the pictures, should be used to acquire these source documents from the variety of DOD and contractor sites that would be expected to produce such documents. Both magnetic media and file exchanges over networks would be used to accomplish the transfer of information.

Once received in source form, personnel at the publishing facility would be expected to make changes to the supplied information for some of the following purposes:

- To put the document into a prescribed format;
- To make uniform the style of writing where a single document is being produced from several contributions;
- To correct spelling, grammatical, and word usage errors;
- To improve the readability and organization of the document;
- To modify or update existing documentation; and
- To modify the supplied pictures to conform to publication conventions regarding line weights, text fonts, and other such graphics arts considerations.

The result of such editing is, again, SGML or ODA logical document files, with their associated CGM picture files, which they themselves may have been modified during the editing process. Where the publishing agency is not the maintenance agency for the document, these revised files should be returned to the maintenance agency as new master files for the document.

In all cases, the publishing personnel should <u>not</u> need to change the substance (that is, the technical meaning) of the documents or pictures. Furthermore, they should <u>not</u> perform any major updates to the publication. Instead, the maintenance agency for the document should make the changes to the master source

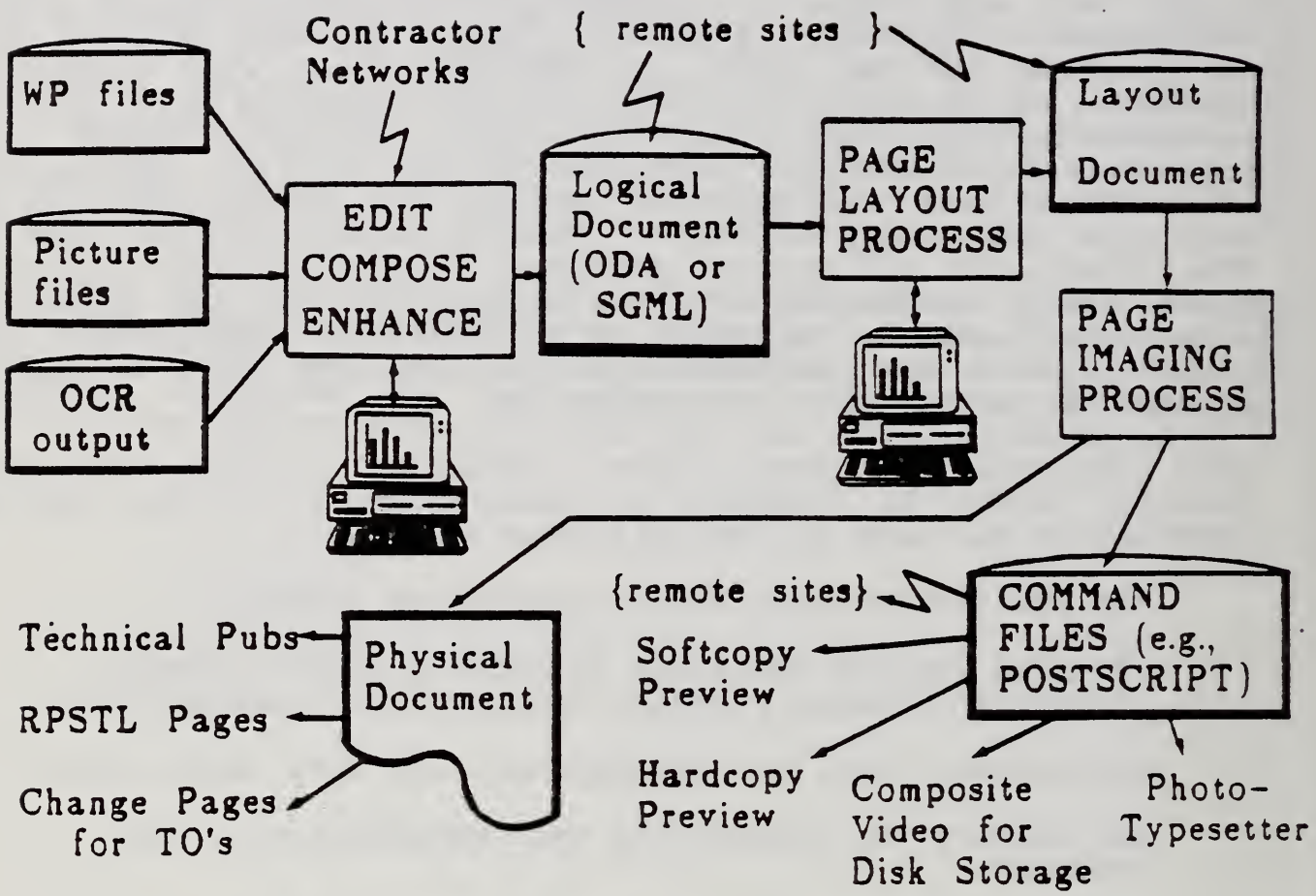


Figure 4-6. Architecture for Publishing

Table 4-4. Publishing.

Input	Process	Output
WP-text	Info. Retrieval	Publications of all sorts (laserprinter)
OCR-text Pictures--Raster --Vector	Exchange with remote sites (e.g., Technology Repair Centers for ATOS)	Typset quality--mixed text & graphics (phototypesetter)
Data tapes CAD Workstation -Vector	Graphics Editing/ Enhancement	Repair parts & special tools lists (RPSTL) pages [APPS]
		Change pages for Technical Orders [ATOS]

material and send new versions of the source files to the publishing agency for republication.

Once the logical document exists in a satisfactory form, a second process--the page layout process--is carried out. Here, publishing personnel make decisions about how the information is presented on the printed page. Such aspects as selection of type face and point size are made at this time. Likewise, formatting decisions concerning paragraph indentation, number of lines between paragraphs, and margin settings are also made at this time. The resulting layout document is no longer considered processable because the location and appearance of every letter and figure on every page has been decided.

The final step--the page imaging process--produces a stream of commands to drive an output device. If, instead of targeting a specific device like an HP LaserJet Plus, the imaging process produces a device-independent command file, such as a PostScript file or a CGI data stream encoding, the command file can be used to produce the final images in a variety of ways:

- The document could be previewed on a CRT workstation to check for errors before typesetting.

- The document could be printed on a lower resolution and less expensive hardcopy terminal to check for errors before typesetting.

- The document could be stored on disk in composite video format for use in an interactive delivery system application (see section 4.3.3.2 below).

- The document could be sent to a phototypesetter or other suitable hardcopy device for production of "camera ready" masters.

- The document could be sent to a remote site for any of the above listed purposes.

At any of the stages in the process, the publishing facility needs to be able to accept from remote sites (other government agencies or contractors) a document of the appropriate format. This implies that each representation of the document should be standardized and that the standards used at each stage should be fully capable of representing <u>all</u> information available at the prior stage. The ODA and SGML standards are being designed to meet these objectives for text; the CGM and CGI standards likewise meet this criteria for pictures. Commercial page imaging "de facto" standards like PostScript and Interpress, too, seem to be sufficiently powerful to meet the needs for a page imaging publication standard, whether generated at the back end of either an ODA or SGML pipeline.

The only role for an API standard--either GKS or PHIGS--in this scheme is during the graphics editing/graphics enhancement stage. Applications written to GKS or PHIGS will be able to easily read in CGM metafiles, modify their contents, and write them out again. Furthermore, the portability and device-independence obtained for the application convey the usual benefits as described in section 4.2 of this report.

4.2.3.2 Interactive Delivery Systems

With on-line, interactive delivery systems there are two phases in their development and use. First, special operators (who typically would be writers and designers with TV or movie production experience) develop a script. Then, they interactively merge a variety of visual and text information, usually available as raster images, to produce a module. Each module consists of a sequence of frames and has a very specific purpose--say, to teach a maintenance engineer to replace a power supply on an on-board computer. Each module will be indexed, so that the module can be easily located by a maintenance engineer when he is trouble shooting on site.

As shown by Table 4-5 and Figure 4-7, once the image database has been created, there are many operations to be supported by the interactive delivery system--each requiring an associated output format consisting of images, forms, and instructions. Voice input and output could also be integrated into such a system.

The performance requirements of such a system are demanding:

- The operator must be able to browse quickly through the database.

- Modules must be found quickly and their initial frames shown within a few seconds after they have been requested.

- The system must support two modes of operation--where the operator has control and where the system has control.

- The system must be able to provide training upon demand and nearly instantaneous help, when needed or requested.

- The operator must be able to initiate work requests and actions.

- The system must be linked to a network to permit access to centralized information, to initiate actions, and to receive status information about work in progress or completed.

- The system must be able to directly monitor, via probes or channel interfaces, the equipment being maintained and tested. Analysis programs must be available to calculate the behavior of the system.

- Display programs must be available to provide test results to the operator in easily understood graphical or tabular formats.

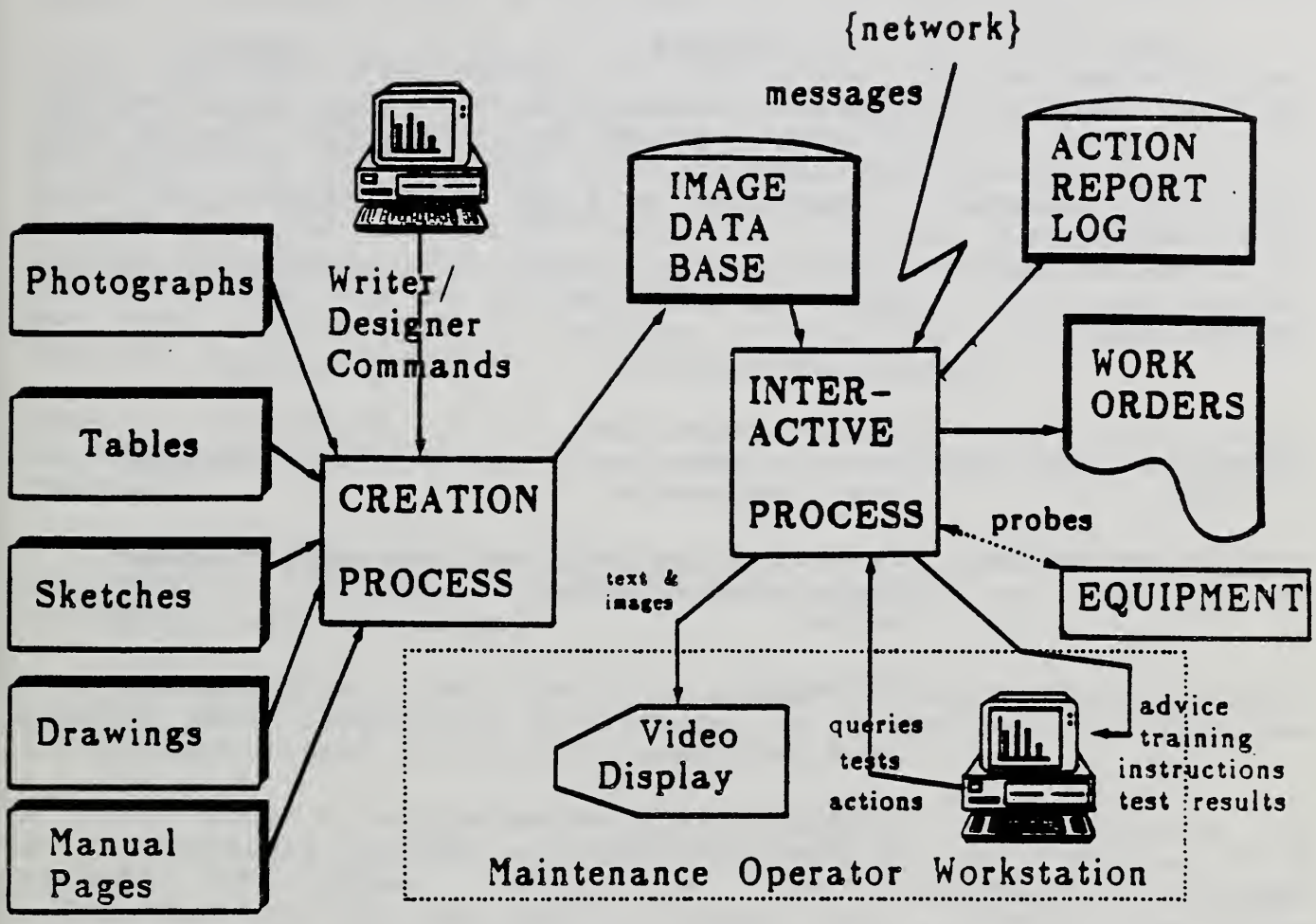


Figure 4 -7. Architecture for Interactive Delivery Systems

Table 4-5. Interactive ("Paperless") Delivery Systems

Input	Process	Output
Raster images	Build & maintain an interactive data base of technical information on disk.	Video images (TV vs. digital frame buffer)
Operator commands [Operators are writers/designers with TV experience.]	Retrieve/search for specific information.	Instructions
	Browse thru DB.	Diagnostic Advice
Report on actions taken.	Damage Assessment. Analyze Data. Test Components.	Work Orders
Orders for parts	File/Organize Reports. Dispatch Orders for Parts.	Training Lessons
	Training.	Messages to peers
	Send messages.	
	Interface with networks, on-board systems.	

In such a system, these requirements imply that, ideally, all information must be stored and be accessible in formatted form; that is, in the final form to be presented to the operator. Otherwise, unacceptable delays may be introduced. Practically speaking, however, such an approach has its own costs.

First, storage requirements for the images can become enormous, especially if coupled to very high resolution displays. Optical disk and TV technology, of course, apply here; indeed, without the existence of such technologies, such application systems as described above would not be feasible. Second, completely formatted images are not revisable and are tied very closely with current hardware. Revisions, corrections, and enhancements to the curriculum are costly snd time-consuming. Upgrading to new and more cost-effective harware is difficult, without making obsolete one's investment in software and images.

From a standards point of view, given the performance requirements, it seems reasonable to adopt the following strategy:

1. During the creation process, keep as much of the input information (drawings, images, text, tables of numbers) in revisable, editable form.

2. Furthermore, the creation process itself should create modules that themselves are stored in a device- and machine-independent format and structure.

3. Only after a module is complete should it be transformed, by an irreversible process applied only once in its lifetime, to a sequence of images. Even then, for infrequently accessed information or for low density images (e.g., forms), the image database should store this information in a device-independent, non-raster-image format.

4. During the display of the information, the stored images will be presented on the video terminal using TV standards for image representation, while the stored text and simpler pictures could use such standards as X3.64 and X3.122 (CGM) to present instructions, forms, graphs, and tables on the operator's terminal.

In this environment, there is little need for the graphics API standards--GKS and PHIGS, except during the creation process. CGI, providing device-independence, would probably be useful during interactive delivery as the interface to the operator's graphics terminal, but not to his TV video display.

4.2.4 Intersystem Relationships

4.2.4.1 Automated Data Repository Projects

Figure 4-8 shows how Automated Data Repository projects might share information and communicate. This architecture diagram is hypothetical in the sense that not all projects shown are planned to communicate information at present. However, the purpose of the figure is to show how information collected, maintained, and output for one project might be used to provide input to another.

Figure 4-8 highlights the data formats that might be used in interchanging pictures. In the figure, the terms "FAX", "GROUP IV", and "Raster" are approximate synonyms, but with the following more subtle distinctions:

- GROUP IV refers to the CCITT standard for facsimile. Only two tone pictures can be represented.

- FAX refers to any facsimile format, whether or not it is in exact conformance with the CCITT standard.

- Raster refers to some future standard for raster images, which should include all the capabilities of FAX but be capable of handling color and gray scale images as well. This standard might be based on an extended CGM, limited to the CGI pixel array element as its only output primitive element.

The CGM and IGES play the central roles in exchanging information between projects--the CGM for diagrams and engineering drawings and IGES for product definitions.

4.2.4.2 Automated Technical Manual Projects

Figure 4-9 shows how the Automated Technical Manual projects might share information and communicate. Like Figure 4-8, this diagram is hypothetical in nature, showing the potential for interchange among such systems.

In this environment, the CGM and Raster formats continue to play an important role, but IGES is no longer needed because none of these projects need to deal with product databases. Rather, they deal with the representations (textual and pictorial) of products. If the product definition needs to be changed, one must go back to the engineering disciplines to make the change. The maintenance, publishing, and training components should not make changes to the product definition without the participation of the engineering function.

Also in this environment, specialized formats like NTSC video and page imaging languages like PostScript and Interpress have a role to play. The formats--sometimes formal standards (CGM, SGML, and ODA/ODIF) and other times de facto standards (PostScript and Interpress)--are needed to provide extremely fast display of frames of information. However, one does sacrifice the device independence that comes from using the formal ANSI standards relating to alphanumeric and graphics standards.

The Text and Office System standards like SGML and ODA/ODIF will play a significant role in the interchange of information among such publishing systems. At present, ODA/ODIF relies on CGM for the incorporation of pictures into documents, while SGML does not yet have any standardized procedure for drawings.

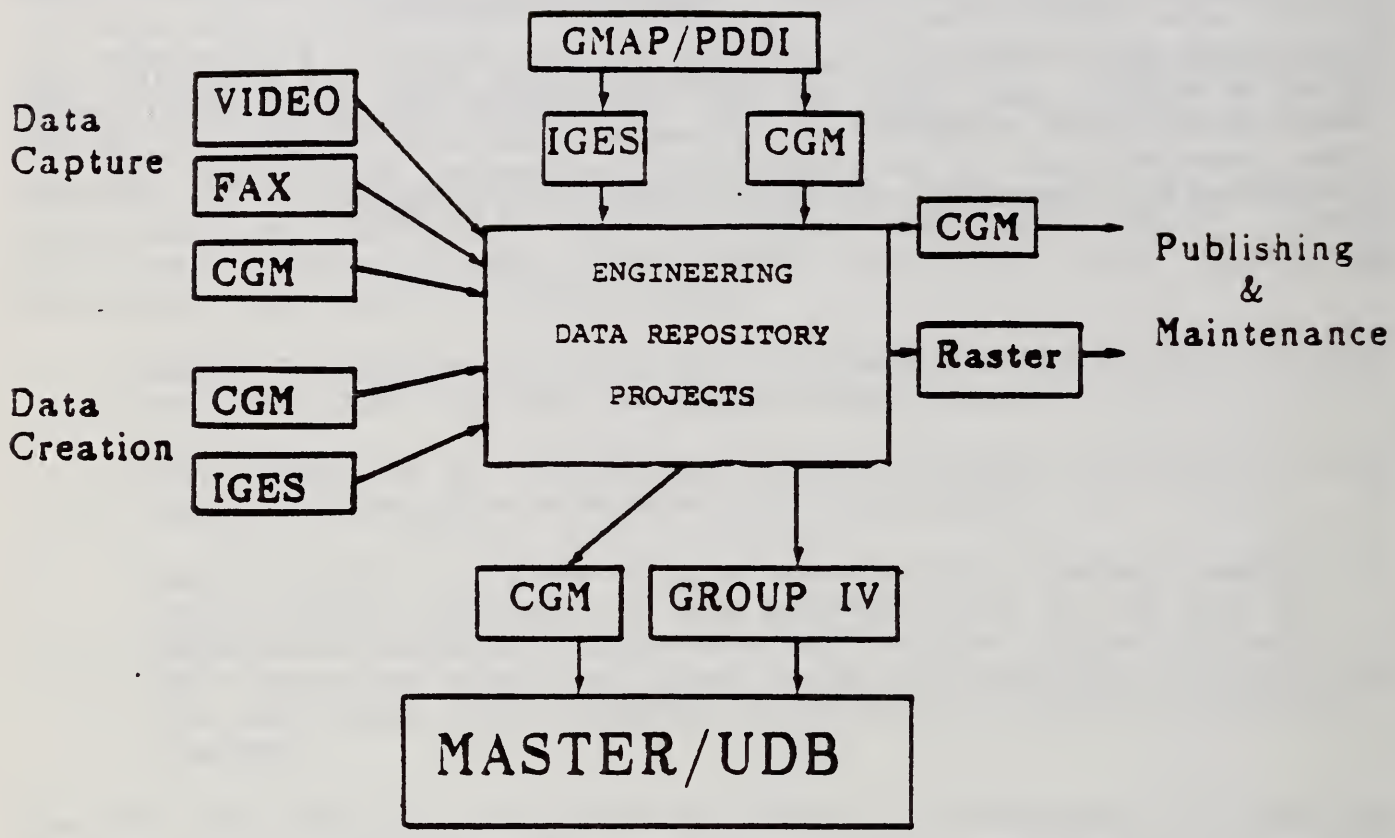


Figure 4-8. Possible Graphics Data Flow Among CALS Data Repository & Procurement Support Projects

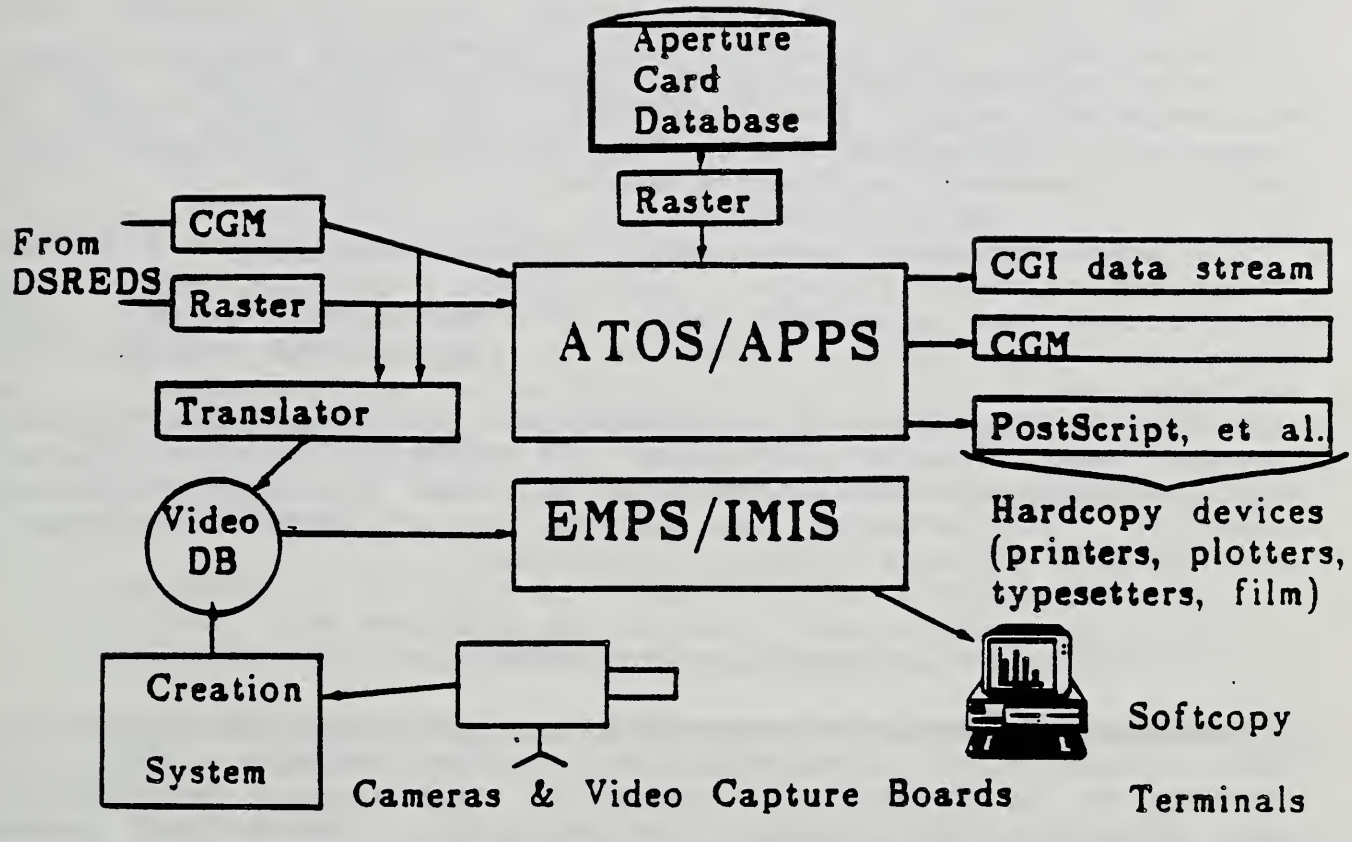


Figure 4-9. Possible Data Flow Among CALS Publishing & Interactive Delivery Projects

4.3 CONCLUSIONS AND SUMMARY

4.3.1 CGM Strengths

The CGM contains device-independent, digitally-encoded vector graphics data. CGM files are easily transferred and displayed on a wide variety of hardcopy devices (dot-matrix, ink-jet, electrostatic, and laser printers, pen plotters, and film cameras); furthermore, they can be easily previewed on an extensive range of softcopy terminals.

The same cannot be said about facsimile and most videotext/teletext formats. Most expect a certain, narrow range of resolutions (sometimes only one!) and either cannot encode color and gray-scale information or require the presence of a certain range of color or gray-scale functionality. Page description languages like Postscript and Interpress, although theoretically device-independent, in practice require generally medium to high-resolution devices in order to produce acceptable results. Furthermore, the performance of these languages is unacceptable in many display situations.

4.3.2 Comparison of Levels of Presentation

Three general levels of representation are available for storing and transmitting pictures: Raster (for example, NTSC video formats or CCITT group IV facsimile), CGM picture descriptions, and IGES/PDES/STEP product definitions. There are several critical qualities by which they can be compared.

Speed of Interpretations. Generally speaking, raster formats are the fastest to interpret, with IGES files being the slowest by many orders of magnitude. CGM files are in between, with the absolute speed affected principally by the speed of the vector to raster conversion. When assisted by hardware, CGM interpretation can be accomplished nearly as rapidly as interpretation of raster files.

Device Independence. Both CGM and IGES files are much more device independent than raster formatted files. Raster files are often tied to one resolution and often do not encode color information.

Size. CGM files are usually more compact than either raster or IGES files. CGM allows geometry to be encoded more efficiently than a full raster representation, as in facsimile, while CGM also need not carry non-visual information as is often present in IGES files.

4.3.3 Picking the Right Level of Exchange

Selecting the proper level of data storage and exchange --raster, CGM, or IGES-- depends upon what one wants to do with the data after one gets it!

Raster. Raster formats are appropriate and adequate if one plans on viewing the image on a limited set of similar devices, whose characteristics (e.g., resolution, color capability) are known in advance and suitable for raster display. Video (e.g., NTSC), facsimile (CCITT Group IV), or videotext/teletext (e.g., NAPLPS) standards are all candidate storage and interchange formats.

CGM. CGM is appropriate for any combination of the following three general situations:

- One wants to view the image on a wide variety of devices, with different color and resolution capabilities. The set of devices may not even be known at the time the metafile is generated.

- One wants to be able to enhance picture qualities (e.g., colors, line weights and styles, font styles and text position) before viewing the final image.

- One wants to be able to compose or overlay several drawings into a single picture for viewing.

IGES. Product data definition files should be used when any of the following conditions occur:

- One needs to exchange product designs, not just drawings or images.

- One needs to use the relationships between entities that comprise the picture for such operations as analysis, simulation, and design rule checking.

- One needs to transfer instructions for viewing the model, not just the views themselves.

4.3.4 CGM as a Raster Format

The CGM can be used to exchange pure raster images using the CELL ARRAY primitive. The standard encodings allow for compression of the image data. <u>Private encodings</u> that follow the same rules as the standard encodings may also be invented if the three standard encodings are inadequate. This method provides a mechanism for getting raster scanned images into, say SGML documents, using the same techniques as for vector-oriented CGMs.

If the image must be scaled to fit the available space, the SGML document processor can be programmed to provide this capability, operating on the CELL ARRAY information contained in the CGM.

4.3.5 Impact of Standards on Software Life Cycle Costs

Four factors result in a **lower overall software life cycle cost** for programs written to graphics standards. Machine independence means that development costs can be shared across many program elements. Device independence allows expensive graphics hardware devices to be shared by many application programs and users, reducing the overall pre-workstation cost. Performance improvements mean that the same host computing environment, augmented with evermore cost-effective graphics display hardware, can have a longer life, thus delaying or even avoiding the need for costly upgrades to more expensive CPUs as the workload increases. Finally, the widespread training in graphics standards concepts means that increases in overall maintenance costs can be held in check. No longer will extensive, specialized, and expensive training be required in order to write and maintain graphics application programs that are written to the ANSI standards.

4.3.6 Benefits of Different Kinds of Standards

The benefits of process-related standards (that is, those for program development--the API standards) are completely independent of whether they are coupled to CGM. The converse is also true: that is, CGM is valuable as a picture storage and exchange standard regardless of whether the metafiles are generated from applications written using GKS, PHIGS, or CGI.

However, due to the strong overlap between functions in the CGM and those contained within the higher-level API standards, it is slightly easier to generate CGM from such applications rather than from non-standard programming environments. Stated more formally, the "semantic match" between the CGI and the other standards is much higher than with most non-standard graphics libraries.

4.3.7 Synergism of CGI and CGM

Because of the deliberate and explicit alignment of CGI and CGM concepts, primitives, attributes, and viewing models, CGM files will be very fast to interpret on new hardware because the CGI functions are going into microcode and even onto silicon. Consequently, in the future, the speed advantage of purely raster formatted images over the CGM will be greatly reduced.

4.3.8 Using ASCII Files

Seven-bit ASCII formatted files are the most transportable of any of the coded formats for picture files, but the IGES ASCII file format is very inefficient compared to the Character-Coded format of CGM. The IGES and CGM binary files are also compact, but are not very suitable for interchange across networks and among heterogeneous computing environments.

5. PLANS AND RECOMMENDATIONS

5.1 RECOMMENDATIONS FOR GREATER COMPATIBILITY AMONG GRAPHICS AND PRODUCT DATA STANDARDS

5.1.1 NBS Recommendations

The principal recommendations concerning compatibilty that NBS endorses are as follows:

--NBS suggests that implementation and user experience be gained before incorporating technical changes to increase compatibility in future versions of the standards.

--A mechanism called "Registration of Graphical Items" is available for the graphics standards; perhaps a similar mechanism would be useful in the product database arena.

--PDES should continue to look to the graphics standards for its presentation entities. These features should be merged with most of the existing IGES entities to form the international standard for the exchange of product data. IGES entities that differ greatly from current graphics practice should be deprecated.

--The exchange standards--IGES, PDES, and CGM--need to tighten up their conformance clauses so that users of the standard are guaranteed a minimum level of service.

5.1.2 Contractor Recommendations

The rest of Section 5.1 contain the contractor's [BON186] own wish list for <u>all</u> that could be done to make graphics standards more compatible with one another and with the product data definition standards. They specify a great many avenues for arriving at more compatible standards. NBS has not had enough time to completely evaluate this list, nor does NBS endorse everything that the contractor states. This list is provided for completeness. The Proposed '87 NBS Statement of Work for CALS prioritizes which areas of this wish list NBS deems feasible for CALS in the next couple of years.

5.1.2.1 Within the Family of Graphics Standards

Not all the differences among the various graphics standards imply that the standards are incompatible and, consequently, would need changing! On the contrary, for example, the CGI was designed to support the requirements of several kinds of graphics systems that might be built on top of it. So elements are

defined very carefully, both to support GKS directly, but also to permit other languages, both device-independent and device-dependent, to use the CGI. Similarly, certain concepts appropriate for PHIGS are missing from GKS-3D.

Examples of differences that are deliberate and don't imply a need for changes to the standards are listed here.

- Background color is set explicitly in CGI/CGM, but is set implicitly in the API standards.

- The absence of window, viewport, and normalization transformations from CGI/CGM.

- The absence of modelling transformation capabilities from GKS/GKS-3D.

- The differences between the segments of GKS and the structures of PHIGS.

- The fact that the segment attributes of GKS are structure elements in PHIGS.

"Near-Term Changes"

The following paragraphs recommend changes to PHIGS, GKS-3D, and CGI that should be studied for inclusion in these draft standards before they are finalized.

For PHIGS:

- Some reasonable meaning should be attributed to the GKS functions ACTIVATE WORKSTATION and DEACTIVATE WORKSTATION in a PHIGS environment. This issue is currently being studied by ASC X3H3 and ISO/TC97/SC21/WG2 as part of the need to resolve PHIGS-GKS compatibility.

- The utility of an explicit, standardized metafile read/write/interpret facility should be studied further. This facility would be provided <u>in addition to</u> the current PHIGS archiving facility. With the present facilities it is awkward to write individual items to metafiles or interpret individual items from metafiles.

- There is a strong effort to harmonize GKS and PHIGS, especially in the area of modelling transformations and the viewing pipeline. These efforts should be encouraged.

For GKS-3D:

- Only those changes that keep GKS-3D consistent with

PHIGS should be made in the near-term. It is vital that GKS-3D be technically frozen, so that products can appear in the marketplace. Future changes should keep GKS-3D current with any changes made to GKS.

For CGI:

-On the output side, only those changes that keep the CGI able to directly support GKS efficiently should be added.

-On the input side, support for new functionality is appropriate. These include AREA and COMPOUND input devices, dynamic association of triggers, and input extent. Other new features for input like user-defined cursors should be heavily driven by requirements generated by Task Group X3H3.6, Display Management.

"Future Changes"

Some of the following paragraphs recommend changes to GKS and CGM that should be incorporated in the next version of the standard. In the meantime, some of these changes should be put forward for registration so that they can be incorporated in implementations in a standardized manner, prior to formal inclusion in the standard. Other changes derive from the current CGI draft and should be incorporated in the API standards if CGI implementation experience shows that these elements are useful and accepted by the user community.

For GKS, GKS-3D, and PHIGS:

-In CGI, one can reset the workstation to its default settings without reinitializing the view surface. This kind of facility would also be useful at the API level.

-The CGI closed figure facility.

-The CGI ability to declare the "maximum color index."

-The CGI ability to specify the "character coding announcer." This permits the program to indicate how the character codes contained in string variables should be interpreted by text processing system within the graphics environment. At present, the API standards combine the concept of character set with the concept of font.

-The CGI output elements--DISJOINT POLYLINE, CIRCULAR ARC, ELLIPTICAL ARC, CIRCLE, ELLIPSE, and RECTANGLE--should be registered, then added.

-The use of the CGI text elements--RESTRICTED TEXT and APPEND TEXT--should be studied, then added in some form to the API standards.

-The CGI permits line width and marker size to be specified in absolute terms as well as with a scale factor Specification of these elements in World Coordinates in PHIGS and GKS might be useful in certain applications.

-Some portion of the CGI raster functionality--bitmaps, drawing mode, auxiliary color, and transparency--needs to be available to GKS and PHIGS application programmers. The new bitmap interior style would be especially useful, when coupled with the ability to create and modify offscreen bitmaps.

-The CGI ability to specify a contiguous block of color indices, and not just one index at a time, should be added to the API standards.

For PHIGS:

-Support for Hidden Line and Hidden Surface Removal. This enhancement should probably be coupled to a general upgrade of PHIGS to handle surfaces and solid definitions.

For GKS:

-Addition of an explicit, standardized mechanism to save and restore segments from GKS Workstation Independent Segment Storage (WISS) should be considered. The mechanism should operate similar to the PHIGS Archive Facility. This facility would allow GKS segments to be used as symbols drawn from a shared "symbol library."

-The PHIGS ability to turn error handling on and off.

-The new GKS-3D function POLYGON SET should be registered and then added to GKS.

-CGI unbundles the concepts of text font and precision to allow independent specification of these aspects. GKS needs to examine application use of font and precision to determine whether changes should be made in future versions of GKS.

-CGI unbundles the concepts of hatch index and pattern index to allow independent specification of these aspects, while in GKS there is just one index that is interpreted depending on the value of interior style. GKS needs to examine application use of hatch index and pattern index to

determine whether changes should be made in future versions of GKS.

-The CGI allows, via the Pattern Size element, patterns to be skewed ("slanted"). This feature is not directly available in GKS and PHIGS; it can occur only as a result of segment or structure transformations.

-The CGI/CGM, GKS-3D, and PHIGS edge attributes--type, width, color, and visibility--need to be registered and then migrated to GKS in the next version. Along with this change, the ASF and ASF3 functions--controlling 13 and 4 elements respectively--in the current GKS should be merged to a single ASF function controlling 17 elements, as is currently specified in PHIGS.

-If application experience is favorable, the PHIGS ability to set the color model (HLS or RGB) should be added to GKS.

-GKS and GKS-3D should be upgraded to support full 3x3 and 4x4 transformation matrices, so as to become consistent with PHIGS and with current practice.

For CGI:

-If application experience is favorable, the PHIGS ability to set the color model (HLS or RGB) should be added to CGI.

-CGI needs to be upgraded to support 3D objects and viewing.

The CGM is a picture exchange specification, while the GKSM has different objectives. ISO/TC97/SC21/WG2 has recognized the desirability of a single collection of metafile elements, coded according to the same principles, that could serve as a basis for different kinds of metafiles. An ad hoc group of experts has recommended that the CGM elements and principles serve as that basis and that extended metafiles be defined to meet the needs currently served by the GKSM. Furthermore, they recommended that the metafiles also be extended to support 3D objects, segmentation, and viewing. PHIGS structures may be accommodated in later versions of the extended metafile. (This very important project is not staffed, as yet, by any US experts and the total manpower available to WG2 is not large. If progress is to be made, additional manpower, preferably from the US, needs to be found.)

In the following paragraphs, a few GKSM anomalies with CGM are highlighted. These would be expected to disappear as the extended metafile project makes progress.

-The specification of precisions of element data types is limited to field length.

-There is no ability to distinguish between the precision of real coordinates and other real numbers.

-There is no ability to distinguish between the precision of indices and other integers.

-There is nothing similar to the CGM's "local color precision" to reduce the storage requirements for CELL ARRAY.

The GKSM permits line, marker, text, and fill representations to be placed in a metafile. In an extended metafile, one would expect to see these elements added to the CGM base set.

5.1.2.2 Within the Family of Product Data Standards

"Near-Term Changes"

No near-term changes are suggested for IGES, because the IGES committee is committed to using the PDES/STEP effort as the test bed for evaluating the utility of new entities. These entities will be migrated to IGES in an orderly fashion, after they have been accepted into PDES.

All near-term suggestions for PDES arise from consideration of the graphics standards and, consequently, are documented in section 5.1.2.3 below.

"Future Changes"

The migration method from PDES to IGES adopted by the IGES Committee is a sound idea. Consequently, no other recommendations need be made in this category.

5.1.2.3 Across Family Issues

"Changes to Product Data Draft Standards"

The following paragraphs document changes needed in PDES and IGES to increase the compatibility of these standards and the graphics standards. As stated at the beginning of this section, these changes are the conclusions of the contractor, and do not necessarily reflect official NBS position.

-PDES should continue to look to the graphics standards for its basic viewing, modelling, and text models. PDES

should continue to avoid all deliberate inconsistencies, especially where matters of taste, rather than technical issues are involved.

-The TEXT FONT definition facility of IGES should be deprecated. This simplistic, STROKE-text facility is nearly useless and is better served by upgrading the whole text model to be consistent with the graphic standards, and by including support for raster and outline character fonts. In general, it is inappropriate for a product definition data base to be specifying the detailed appearance of the text fonts used in rendering a view or a drawing. These matters are best left to the graphics support system, including the graphics device itself.

-Some of the IGES entities (e.g., SECTION) allow for the specification of the details of the interior hatch pattern to be used to fill the section area. Where standard practice exists (that is, where special fill styles, say narrowly-spaced cross-hatched lines at 45 degrees and 135 degrees, by convention mean certain things, say an insulated interior wall), the use of the standard fill pattern should be provided <u>by reference</u> to the other standard practice rather than <u>by specification</u> of the details of the pattern in the product definition file.

-IGES and PDES should adopt conformance rules for translators (or interpreters). All translators should be required to process a minimum set of entities. Furthermore, for each entity type--required or optional--some minimum standards for fidelity of processing should be specified in the standard.

-IGES and PDES should adopt a 4x4 matrix, allowing for the specification of arbitrary modelling and viewing transformations. Existing IGES data bases need not use all the matrix elements to specify their current views.

"Changes to Graphics Draft Standards"

The addition of several facilities to the graphics standards would greatly improve the ability of these standards to efficiently represent the drawings and views implicit in IGES and PDES files. These additions are described in the following paragraphs.

-Support for full conics including hyperbolas and parabolas.

-Support for splines, including at least one of the parametric spline and rational B-spline representations.

-Support for surface definitions, including surfaces of revolution and cylinders.

-Support for a ROUNDED RECTANGLE output primitive.

-Support for many new line types, including some or all of the 11 forms of "arrows" defined in IGES through registration and subsequent standardization.

-Support for the CENTERLINE symbol as a new standardized marker type.

-Addition of facilities to more directly control and specify such features of text strings as subscripts, superscripts, and fractions. At present, these features can be generated only by using the more primitive APPEND TEXT and by changing associated text attributes like CHARACTER HEIGHT, CHARACTER SPACING, and TEXT ALIGNMENT.

-In IGES/PDES the slant of the text is independently specified from the type face (e.g., Helvetica Italic Bold). The CGI allows, via the Character Orientation element, text characters to be skewed ("slanted"). This feature is not directly available in GKS and PHIGS; it can occur only as a result of segment or structure transformations.

-In PDES, continuous text alignment is used to align multiple text strings. This feature is also available in CGI/CGM. It should be added to GKS, GKS-3D and PHIGS.

-The six predefined IGES SECTION entity patterns not corresponding to standardized CGI/CGM patterns should be registered.

-Support for user-defined line types.

-Support for multiple color tables.

-The CGM should adopt conformance rules for translators (or interpreters). All translators should be required to process a minimum set of elements. Furthermore, for each element--required or optional--some minimum standards for fidelity of processing should be specified in the standard. That is, the current guidelines in Appendix D of the CGM standard need to be improved and promoted to official parts of the standard.

5.2 RECOMMENDATIONS FOR GRAPHICS STANDARDS VALIDATION

Based on the extensive analysis of the European graphics validation suite and experience installing and executing the tests in a typical US graphics environment, NBS makes the following recommendations to CALS:

1. The data structure tests are of too low quality to be useful for validating GKS implementations for DOD purposes. Rather than expending the resources necessary to correct deficiencies in the tests, and due to the fundamental flaws described in [CARS86], it is recommended that the operator interface tests be expanded to include testing of the most important data structures.

2. The error tests are of higher quality than the data structure tests. If fundamental flaws are corrected, the tests could be used to provide a useful validation of the error handling of GKS implementations. A moderate effort would be required to upgrade the routines. The error tests should remain a separate set of tests since they are difficult to integrate with the operator interface tests.

3. The operator interface tests are of definite value in validating GKS implementations for DOD use. A moderate effort would be required to correct programming errors in the tests and to make them more device-independent. As stated in (1) above, the test could be expanded to include tests of the most important GKS data structure with little impact on their run-time efficiency.

4. The test programs are available only in FORTRAN. Since DOD environments are likely to use Ada, consideration should be given to converting the tests to that language. This would be fairly straightforward due to the simple structure of most of the test programs and subroutines, but a time-consuming process.

5. Consideration should be given to restricting some of the options available in GKS when it is used in a DOD environment. Additional constraints should also be placed on the "correct" interpretation of many of the effects which GKS defines to be "device- or implementation-dependent." If such constraints are defined and promulgated, the test suite should be expanded to test for them.

6. The test suites do not test the Computer Graphics Metafile (CGM) and only test the GKS Metafile (GKSM) in a cursory way. Due to the importance of graphical

metafiles for the CALS program, work should start at once to develop a test suite for the CGM.

5.3 POSSIBLE SHORT AND LONG TERM SOLUTIONS

The contract report [SPAR86] detailed a proposed short and long term solution for the use of graphics within the CALS environment. It is reprinted in Appendix 6 to give DOD a perspective of how an expert in graphics standards foresees a solution to the raster vs. vector problem in CALS. NBS does not endorse this scenario, but rather specifies the need within the draft plan to assess current raster to vector technology to get a better handle on this CALS problem.

5.4 RECOMMENDATIONS FOR USING COMPUTER GRAPHICS STANDARDS

This section is devoted to Dr. Bono's recommendations from his second CALS contract report [BON286]. Again, these recommendations are Dr. Bono's, and represent his view of CALS priorities as it regards incorporating graphics standards into the CALS environment. Since he does not have to limit his priorities based on available NBS personnel and resources, his is a rather ambitious view of what should be done. As previously stated, they are all reprinted here for completeness, and to give DOD CALS a perspective on all the choices that NBS had to weigh in coming up with the priority list contained in the FY87 Statement of Work, or CALS Draft Plan.

5.4.1 Accelerated Development Work

Work on the Computer Graphics Extended Metafile (CGEM) should be given a high priority.

There is a wide gap between the relatively simple representational capabilities of CGM and the very rich capabilities of product data definition standards like IGES. Conversely, there is also a wide gap between the compact size and straightforward processing of CGM files and the enormous size and convoluted processing of IGES files. The CGEM is a proposed ISO project to add functionality to CGM in a staged, upwardly-compatible manner. The stages would include, at a minimum, adding segments and 3D viewing. Facilities for more extensive hierarchical support, new output primitives and attributes, and symbol libraries could also be considered.

A mechanism for registering special CGM ESCAPES and APPLICATION DATA elements should be built into MIL-STD-1840.

NBS will need to determine if DOD-specific ESCAPES and APPLICATION DATA elements are needed to augment the CGM for its use with MIL-STD-1840. The registraion procedures could be modelled after those being developed within ISO for the Registration of Graphical Items.

For MIL-STD-1840, NBS needs to determine what encoding(s) is (are) suitable for CGM.

All three encoding formats--character-coded, binary, and clear-text--have their uses, but were designed for different purposes. Depending on the purpose of MIL-STD-1840, some or all of the standard encodings of CGM should be included by reference.

NBS should form a joint IGES/CGM team to identify needed CGM ESCAPES.

There are many situations in which IGES is being used but for which IGES was not designed. The current CGM may not be able to be used either, due to the absence of just a few items. Identification and specification of these needed, missing elements by the NBS team would permit the CGM to assume more roles where its compactness and simple processing would allow more efficient and less costly use of computer and human resources. Generally useful new elements could be proposed for registration, either within the more limited scope of MIL-STD-1840 or in the wider ANSI and ISO arenas.

Work on the Programmers Hierarchical Interactive Graphics System (PHIGS) Standard should proceed according to the current published schedule.

A recent survey of attendees (48 responses out of 150 attendees) at the PHIGS Tutorial course given at the SIGGRAPH '86 Conference provided the following insights:

- about one-half the respondants supported aerospace, engineering, science, government, or military applications.

- over 80% of the respondants need PHIGS within one year.

- nearly 70% said that PHIGS was the best suited API standard for their purpose.

- of the users, 24 of 32 (75%) stated that they needed exact or almost exact PHIGS conformance.

Although these results are based on a small and biased sample, Dr. Bono believes they correctly reflect DOD's need for a PHIGS

API standard. Because of the overwhelming life cycle cost benefits that accrue from API standards, PHIGS should be developed as soon as is practical, according to the published ISO schedule.

5.4.2 Urgent Requirements for Validation

Validation of API-level Standards Should Continue.

The beneficial impact of using API-level graphics standards to keep software life cycle costs under control is so great that NBS should accelerate its efforts to validate GKS implementations for CALS use (see section 5.4 above). Planning should commence for the validation of GKS-3D, PHIGS, and CGI implementations, when these draft standards are approved. Furthermore, as shown by [CARS86], the DOD cannot rely on test suites developed abroad, although these efforts may be used as a starting point for US validation work.

Validation of CGM Interpreters is Urgently Required.

The CGM standard places conformance requirements on CGM files themselves. In brief, there are rules about what elements may appear in a CGM, in what order these elements may be placed, and the relationship of pictures one to another. Furthermore, fully conforming CGMs must be coded with one of three standardized encoding formats.

Although stated in terms of the CGM file itself, these conformance requirements may be viewed as placing implied requirements upon the metafile generator. Consequently, validation procedures developed for the CGM standard will in fact also serve to validate CGM generators.

However, there are no conformance requirements on CGM interpreters. The appearance of the pictures produced by interpreting a conforming CGM is not specified by the standard. (The standard does offer guidelines, but these are not mandatory and, consequently, will not be tested by any CGM validation effort.)

For CGM to serve the CALS program well, there is a need for a validation suite for CGM interpreters. The validation suite would consist of three major parts: (1) conforming CGM files in each of the three standardized encodings; (2) expected pictures, rendered on a variety of output media, illustrating the range of allowable differences; and (3) a written description of the expected output, detailing exhaustively the range of allowable differences in the resulting image.

The value of this effort would be greatly enhanced and the difficulty greatly reduced, if recommendation R34 were also adopted concurrently with this recommendation.

5.4.3 Accelerated Implementation

Automatic digitizers should output CGM files.

The output from automatic digitizing equipment takes the form of primitives and attributes that correspond closely to those of CGM. The DOD should encourage vendors to, in fact, adopt CGM--perhaps with a few needed extensions registered as ESCAPES or APPLICATION DATA elements--as their principal, non-proprietary data format.

Conversion to digitized drawing databases should occur as quickly as funding allows.

The urgent problem of converting massive quantities of drawings from their raster (image) form (e.g., on aperture cards) to a vector format (e.g., CGM) needs to be solved <u>only</u> for the existing database. It is equally urgent that CALS <u>not</u> <u>increase</u> the volume of non-digitized drawings. Consequently, building new systems based on digitized, vector geometry <u>now</u> will pay dividends in the future, to the degree that CALS can slow down the increase in volume of non-digitized drawings and data.

The CGM should be targetted for accelerated implementation.

Given the important contribution that the CGM can make to achieving the CALS goals, the DOD should focus resources on encouraging commercial implementations and DOD use of the CGM. Availability of fully validated CGM interpreters from a variety of suppliers would stimulate wide usage of the CGM.

The DOD should promulgate CGM Implementors Guidelines for CALS.

Not all CGM interpreters will produce the same picture, even when given the same CGM file and the same target output device. (See recommendation R22 above.) For CALS, CGM interpreters should be required to meet guidelines that specify at least the following:

- a list of required elements;
- minimum sets of functions and features;
- requires simulation for all unsupported elements;
- specific behavior for some (or perhaps all) of the aspects of the CGM standard that are currently left implementation dependent.

These guidelines (sometimes called Application Profiles) could be

developed at a series of CGM implementers workshops and technology demonstrations, perhaps organized by NBS. Appendix 7 contains an initial draft of a CGM Application Profile being recommended for inclusion into the TOP industry standard. TOP requirements may not exactly overlap CALS requirements, so CALS needs to do its own assessment of needs. Nevertheless, unnecessary deviation from the TOP and other Application Profiles that may emerge (such as from ISO TC97/SC18 for incorporating pictures into office documents) should be avoided.

5.4.4 Related Standards

SGML needs to be able to import CGM files to create mixed text and graphics documents.

Hooks are needed in SGML to permit the importing of CGM pictures via external references to files and with "presentation attributes" like those being recommended for the Geometric Graphics Content Architecture of the ODA/ODIF standard (ISO 8613/8). The presentation attributes control position, mirroring, and orientation of the picture; presence and appearance of any border; and size and aspect ratio of the displayed image within the block allocated for the picture.

DOD should continue to insist upon the complete separation of function (semantics) from format (syntax).

The graphics standards have been carefully crafted to allow multiple, alternate syntaxes (either encodings--as in CGM--or language bindings--as in PHIGS and GKS) for a single functional specification. This permits a syntax to be selected according to criteria important to a given application area, without sacrificing the benefits of standardizing the functions.

Other standards groups have not been so careful with their specifications. As a result, alternative standards--with different semantics, but solving the same problem--often get invented and promulgated because of dissatisfaction with syntax. For example, the French alternative--and threat to IGES--arose to a significant degree because the initial format of IGES files specified in the standard was so inefficient and represented obsolete technology. By the time IGES remedied the problem, a competitor had been invented. Naturally, this competitor also has many functional "improvements" over--and incompatibilities with--the original IGES specification.

5.5 PRIORITIES FOR KEY PLANNING ELEMENTS

As a result of NBS/ICST studies of CALS requirements over the past few months, and the schedule that has been developed for CALS, CGM has been identified as the Computer Graphics interchange standard of most immediate utility to CALS. Thus, all priorities are based on getting CGM to the level of completeness that CALS requires. NBS/ICST studies of CALS requirements have revealed that CALS applications currently have made little or no use of microcomputer graphics, so this has been given no priority for near term CALS. There is currently so much work to be done on developing and implementing database management standards and capabilities within the CALS environment, that interfacing graphics standards with them has been deemed premature for the near term. In addition, no work has been formally started in the standards communities to tackle this problem.

The Proposed FY87 NBS Statement of Work in support of the DOD CALS program represents the Draft Plan for CALS work over the next two years. Contained within it are the recommendations and priorities for the incorporation of graphics standards into CALS. Since the CALS Architecture work is likely to drive any changes to this plan, NBS is not going to draft a plan separate from this proposed Statement of Work in the area of Computer Graphics Standards.

To briefly restate those plans and priorities here, they include:

1. Develop a link between IGES/PDES data files and CGM/GKS/PHIGS picture files. CAD/CAM packages, which produce IGES files, provide input to both automated technical manual and engineering data repository systems. To minimize storage and processing overhead and maintain required system performance, CGM should be used to transfer graphical pictures within and across these systems. A mechanism must be developed which supports the transfer of data in IGES format to CGM.

2. Extend CGM conformance definitions to include "generators" and "interpreters" of metafiles. Currently, conformance requirements in the CGM refer to the syntax of the metafile, not to programs which generate metafiles and read metafiles. Conformance criteria are needed for these programs to ensure that the complete graphical image is ported between devices.

3. Update Air Force MIL-STD 1840 to incorporate full CGM implementation for the transfer of graphics pictures. To support CALS requirements, this MIL-STD must provide for the use of CGM in the transfer of graphics pictures.

4. Inject CALS requirements for extended CGM. Work is just beginning to develop a more powerful and consolidated metafile

for graphics picture transfer. This metafile would allow much more substantial modifications to be made on pictures being transferred and would be more compatible with existing graphics standards (GKS, GKS-3D, and PHIGS). It is imperative that CALS requirements with respect to picture modification be defined and input to this effort in its early stages of development.

5. Identify CALS requirements for CGM registration. A registration mechanism is being put in place by ISO to allow extensions to existing graphics standards. As has been stated, NBS/ICST has been chosen as the Registration Authority for this task. Through registration, various primitives (geometric shapes) can be added to CGM. A discussion of areas where registration could be helpful appears in Section 5.1 of this report. This mechanism should be utilized to ensure that all geometric objects which are displayed graphically in CALS can be represented by the CGM.

6. Accelerate development of CGM validation routines which are underway in Europe. Validation routines must be in place for CALS to accurately know if vendor implementations really conform to the CGM standard.

7. With the extended CGM definitions of priority (2) above delineated, develop a plan for additional CGM conformance tests needed to validate generators and interpreters of CGM metafiles. These validation tests must also be in place for CALS to accurately know if vendor implementations really conform to the extended CGM definitions for generators and interpreters as a result of priority 2 above.

8. Conduct an in-depth study of raster-to-vector conversion technology to include a technical assessment, a comparison of the vector output of this technology with the vector outputs of CGM and IGES, and recommendations for CALS. All of the graphics standards, including IGES, require vector format to be utilized effectively. The availability of this technology must be assessed and alternative strategies developed if raster-to-vector conversion is not available in a timely manner.

In addition, work in the graphics standards area for CALS is incorporated in a number of other tasks in the Proposed FY87 Statement of Work, in such areas as defining the core CALS standards package, holding DOD/NBS/industry workshops for the presentation and development of core CALS standards specifications, and in developing the CALS Architecture.

5.6 FINAL CONCLUSION

This section of the NBS final report has detailed all the work done in the graphics standards area, both by NBS and by the contractors hired to help NBS in identifying CALS needs in terms of graphics standards. It has also explored areas where the graphics standards could or should interact with other standards which have been identified as requisite for CALS, such as IGES, PDES, and SGML, and how such interaction could take place. This section of the report has also specified some "architectures" for describing CALS application areas in terms of the use of standards, in particular graphics standards. This work may be of help in the planned architecture work for FY87.

The overall final result of this section of the report is that CGM has been identified as the graphics standard of most immediate benefit to DOD CALS. It is currently an ANSI approved standard, and is shortly to become a FIPS standard. The areas of work in the Proposed Statement of Work are, therefore, geared to making CGM a complete reality for the CALS core specification, including CGM's inclusion into Air Force MIL-STD 1840, work on conformance tests for CGM implementations, and identification of CALS requirements for CGM registration.

REFERENCES

The official standards documents used as source material for this report include:

Computer Graphics Interfacing Techniques for Dialogues with Graphical Devices (CGI), ISO TC97/SC21 N1179, First Working Draft, May 9, 1986.

Computer Graphics Metafile for the Storage and Transfer of Picture Description Information (CGM), ISO DIS 8632 Parts 1 through 4, 1 November 1985; also, dpANS X3.122-1986 (to be published in the fall of 1986).

Graphical Kernel System (GKS), ISO 7942, 15 August 1985; also, ANSI X3.124-1985, June 24, 1985.

Graphical Kernel System for Three Dimensions (GKS-3D), ISO 2ND DP 8805 (ISO TC97/SC21 N853), 1 October 1985.

GKS Metafile (GKSM), Annex E of ISO 7942, 15 August 1985; also, Appendix E of ANSI X3.124-1985, June 24, 1985.

Initial Graphics Exchange Specification (IGES), Version 3.0, National Bureau of Standards Report NBSIR 86-3359, April 1986.

Presentation Entities for the Product Definition Exchange Specification (PDES), ISO TC184/SC4/WG1 N53, November 4, 1985.

The STEP File Structure, ISO TC184/SC4/WG1 document 4.2.2 (working paper), February 7, 1986.

Programmer's Hierarchical Interactive Graphics System (PHIGS), ISO/TC97/WC21 N819, 4 November 1985; also, dpANS X3.144-1985, November 1985.

These documents are available from a variety of sources:

The published ANSI standards (viz., GKS) and the ISO documents (CGI, CGM, GKS, GKS-3D, PHIGS) can be obtained from ANSI, 1430 Broadway, New York, NY 10018.

The draft ANSI standards (CGI, CGM, GKS-3D, PHIGS) can be obtained from the X3 Secretariat, CBEMA, 311 First Street, Suite 500, Washington, DC 20001.

The IGES and PDES documents can be obtained from the IGES Committee Chairman, Mr. Bradford Smith, National Bureau of Standards, Gaithersburg, MD.

In addition to the official standards documents, the contract reports referenced include:

[BON186] Bono, Dr. Peter R., "A Comparison of Standards Relating to Graphics Picture Exchange - A Technical Analysis of the Compatibility of Product Data Definition Standards and Graphics Standards," June 1986.

[BON286] Bono, Dr. Peter R., "A Framework for How the CALS Program Should Utilize Computer Graphics Standards," October 1986.

[CARS86] Carson, Dr. Steve, GSC Associates Inc., "Evaluation of European Graphics Validation Suite," July 1986.

[SPAR86] Sparks, Madeleine, System Development Corporation, "Interchange and Graphics Standards for CALS Applications - Review and Analysis of CALS-Related Requirements for Graphics Standards," July 1986.

GLOSSARY

blind interchange

A mode of interaction between a service and its client in which the client requests services without recourse to interactive negotiation before use or acceptance of response during use. This mode of interchange may result from the service's inability to respond, the client's lack of interest in listening, or limitations in the communications path between client and service.

character set

The set of displayable symbols mapped to individual character codes in a text string. A character set is independent of the font or typeface.

device-dependent

A system or portion of a system that contains logic, algorithms, or data that are consistent with the behavior of a specific graphical device.

device-independent

A system or portion of a system that contains logic, algorithms, or data that do not require nor represent knowledge about the behavior of any particular graphical device.

device coordinates

The coordinates native to a device; device-dependent coordinates; physical device coordinates.

escape functions

Graphical functions that describe device-dependent or system-dependent elements used to construct a picture, but that are otherwise not standardized.

external functions

Functions present in some graphics standards that communicate information not directly related to the generation of a graphical image.

metafile

A mechanism for retaining and transporting graphical data and control information. This information contains a device-independent description of one or more pictures.

metafile generator

The process or equipment that produces a metafile.

metafile interpreter

The process or equipment that reads a metafile and interprets the contents to produce again the picture represented in the metafile.

modelling coordinates

Local world coordinates tied to some object being modelled by the client and viewed by the graphical system.

negotiation

The interchange of inquiry and response by which a client of a set of services determines which capabilities are provided by the service and what the characteristics of the service are.

normalized device coordinates (NDC)

Coordinates specified in a device-independent coordinate system, normalized to some range (typically 0 to 1).

prior agreement

A process whereby the generator of a metafile and the recipient of the metafile come to some understanding regarding the content or format of the metafile, that understanding <u>not</u> being recorded in the metafile itself. In a blind interchange environment, prior agreement can be used to overcome limitations of exchange standards.

segment

A collection of graphical functions that can be manipulated as a unit. Once functions are grouped into segments, they are referred to as segment elements.

world coordinates

Coordinates specified in a device-independent coordinate system, whose units are selected by and are meaningful to the client.

APPENDIX 1

COMPUTER GRAPHICS STANDARDS

APPENDIX 1

COMPUTER GRAPHICS STANDARDS

1. PURPOSE OF VARIOUS GRAPHICS-RELATED STANDARDS

1.1 Position Within a Reference Model

Standards codify the exchange of information across an interface between two functional units and specify what is to be exchanged, but not how the functional units should carry out their operations. Figure 1 depicts a reference model showing the various interfaces in a graphics application process. This process may receive information from an external product definition database over the product definition interface. This application process interacts with physical graphics devices and operators via a computer graphics environment, which in Figure 1 has been partitioned into a support package level and a workstation level. The interfaces significant for graphics standards are depicted.

Figure 2 shows the current standards under development, positioned at their appropriate levels. This figure highlights the two interfaces that are central to graphics standardization: the application programmer interface, or API, and the virtual device interface, or VDI. Standards at both interfaces provide device independence for the user of the standard. That is, the user can deal with one or more abstract ("virtual") graphics devices with a full range of input and output capabilities. The "messy details" of the particular hardware capabilities of any particular graphics device are hidden from the user. Instead, implementations of the standards must emulate any required facilities not directly supported by the hardware. Furthermore, the implementations mask the peculiarities of the particular command sets used to communicate specific orders to the graphics devices.

1.2 The Application Programmer Interface

To exchange pictures among diverse applications and across separate programming environments, information can be captured at an interface and placed in a graphical metafile, a formatted disk or magnetic tape file containing graphical commands and data. These files can be transmitted over telephone lines and computer networks to be stored and processed by the recipient.

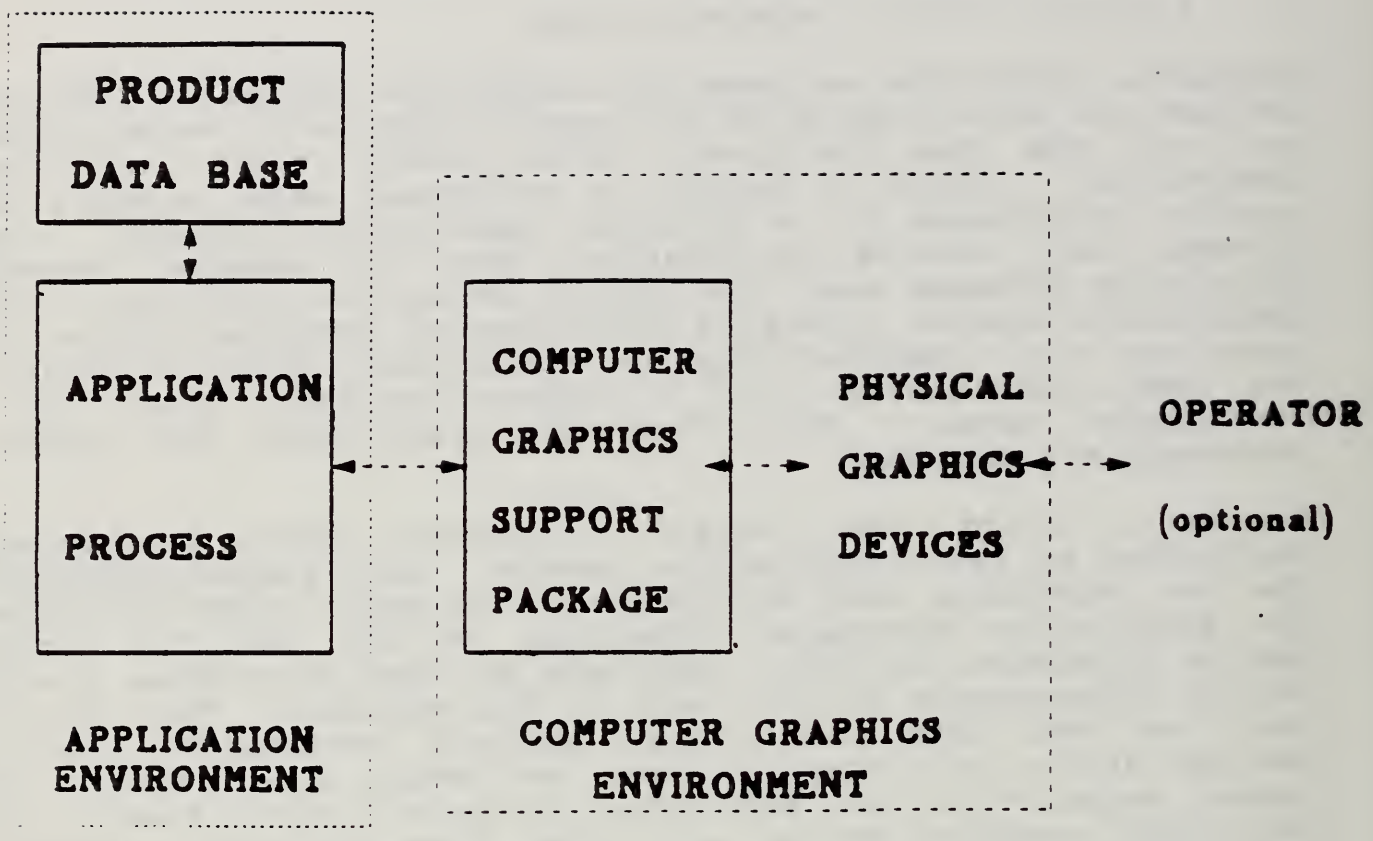

PRODUCT
DATA BASE
APPLICATION
PROCESS
APPLICATION
ENVIRONMENT
COMPUTER
GRAPHICS
SUPPORT
PACKAGE
PHYSICAL
GRAPHICS
DEVICES
OPERATOR
(optional)
COMPUTER GRAPHICS
ENVIRONMENT

Figure 1.

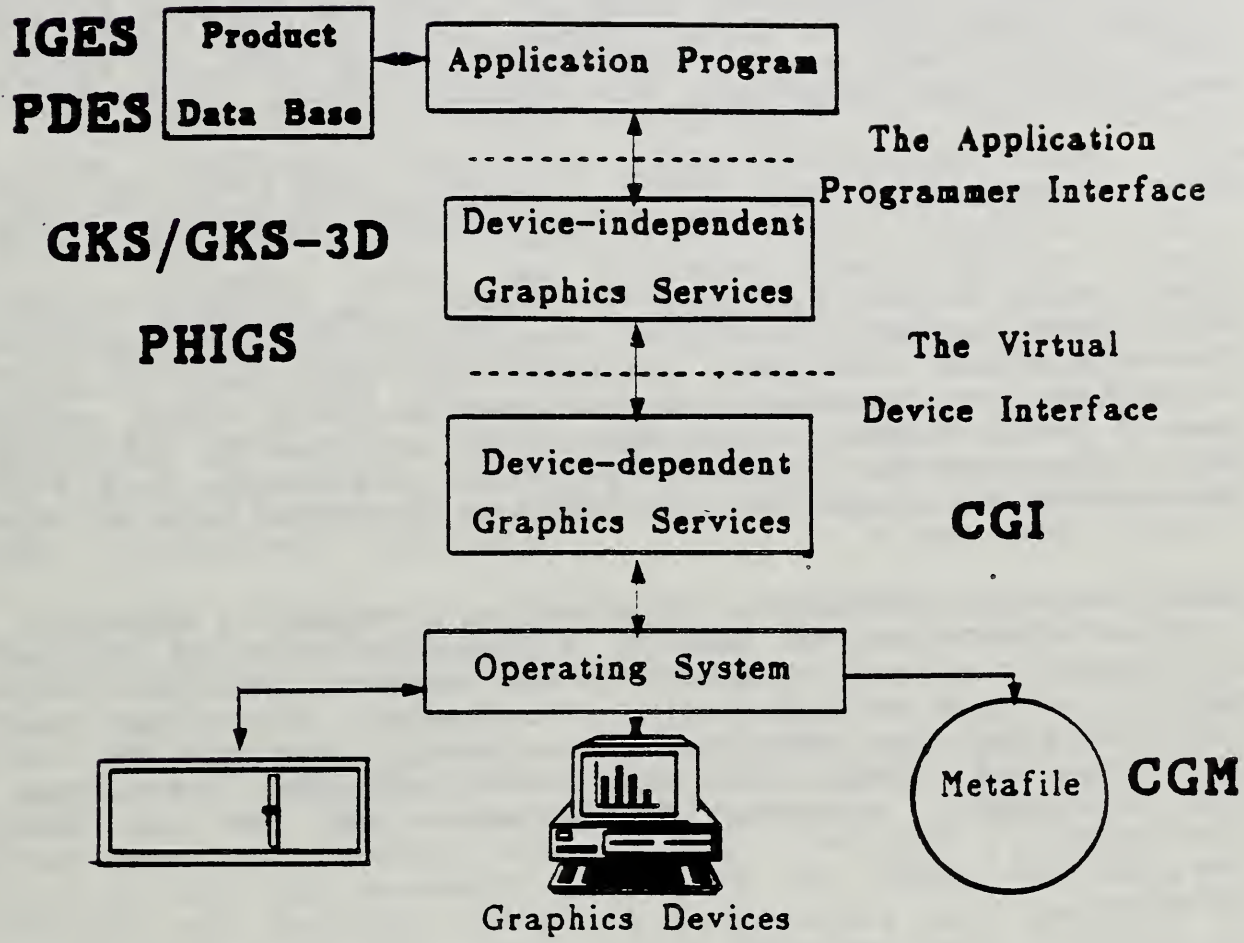

IGES
PDES
Product
Data Base
Application Program
The Application
Programmer Interface
GKS/GKS-3D
PHIGS
Device-independent
Graphics Services
The Virtual
Device Interface
Device-dependent
Graphics Services
CGI
Operating System
Metafile
CGM
Graphics Devices

Figure 2.

The API is represented by three major graphics standards projects: GKS, GKS-3D, and PHIGS. These API standards are typically implemented as a collection of external procedures or subroutines that a programmer can link with his application code to obtain graphical input and cause pictures to be displayed on graphical output devices.

The API standards are not directly suitable for picture exchange. However, each standard has an associated storage mechanism (an archive file or graphical metafile) that can be used to exchange graphical information among systems using the same standard. The API standards are briefly described in the following paragraphs. Their associated metafiles are described in later sections.

GKS. GKS consists of nearly 200 user interface routines that give a programmer the ability to create graphical output and accept graphical input from a wide variety of graphical devices. These include black-and-white and color displays, printers, plotters, and camera systems of varying resolutions, as well as mice, data tablets, keyboards, joysticks, and digitizers. Only 2D primitives are used to describe pictures, although 3D renderings can be created by the application by first performing the 3D to 2D mapping itself before calling the GKS functions. A GKS implementation typically provides programming access to GKS from several higher-level programming languages such as Fortran, Pascal, Ada, and C.

GKS's purpose is to allow the creation of <u>views</u> of objects. Each view is described to GKS by a succession of primitives and attributes that may be grouped into segments for later viewing, without the client having to respecify the primitives and attributes in the segment. As a whole, segments may be made invisible and highlighted, translated, rotated, and scaled, but the individual contents of segments may not be modified. Consequently, GKS is a pure viewing system; GKS does not keep anygraphical model or graphical database for the application program.

GKS-3D. The project to specify extensions to GKS for defining and viewing three-dimensional wire-frame objects is well underway. Like GKS, GKS-3D is restricted to viewing objects: nomodelling is performed by a GKS-3D implementation. This limitation in GKS and GKS-3D keeps the size, complexity, and cost of the implementation down and meets the needs of at least 80% ofthe graphics applications today, including much of the needs of business graphics, statistical and engineering graphics, project management, and mapping.

PHIGS. The Programmer's Hierarchical Interactive Graphics System is an emerging standard specifying an application programmer's interface to a rich, device-independent graphics environment.

PHIGS is designed to support such important applications as CAD/CAE/CAM, command and control, molecular modelling, simulation, and process control. PHIGS emphasizes thesupport of applications needing a highly dynamic, highly interactive operator interface and expects rapid screen update ofthe complex images to be performed by the display system.

PHIGS provides all the viewing capabilities of GKS-3D in a compatible manner, but, in addition, PHIGS supports the creation,modification, and viewing of a geometric model, which is maintained by the PHIGS implementation. Called the Central Structure Store, PHIGS elements are structured into hierarchies, with structures calling other structures and with offspring structures inheriting attributes from parent structures. Once created, or while being created, PHIGS structures can be marked for displaying on one or more workstations.

The current state of technology dictates that the initial implementations of PHIGS will be designed to run on nothing less powerful than a superminicomputer, using a high-performance display. Until fast, floating-point hardware is commonly and inexpensively available, until processor cycle times improve substantially, and until internal word sizes and data path widths increase to 32 bits or more, PHIGS is unlikely to be implemented, in toto, on personal computers.

The principal purpose of the API standards is to provide portability for the application program across a wide range of operating systems, programming languages, and interactive graphics devices. Consequently, programs written to an API standard at one facility can be exchanged with another facility and used with only minor modifications needed to tailor the software to the implementation differences allowed by the standard.

Furthermore, as hardware CPUs and peripherals are upgraded and replaced, software written to an API standard will survive and need not be rewritten. Indeed, the software performance should improve, assuming that the new hardware is more capable than the old hardware.

1.3 The Virtual Device Interface

The VDI is internal to the graphics system and concerns system programmers, independent software vendors, peripheral device manufacturers, and graphics controller board and graphics chip makers. These clients require device independence without sacrificing performance.

The CGI (Computer Graphics Interface) standard is designed to specify the exchange of information at the VDI, while the related

CGM (Computer Graphics Metafile) standard serves to capture the descriptions of pictures at the level of the CGI. These two standards are described more fully in this and the next section.

CGI. The Computer Graphics Interface is a standard functional and syntactic specification for the exchange of device independent data and associated control information between systems with graphical functional capabilities. These systems may be peer graphics systems or may be device dependent graphics device drivers.

The CGI defines an idealized abstract graphics device capable of accepting input and generating, storing, and manipulating pictures. It contains elements for generating graphical primitives; controlling the appearance of graphical primitives; inquiring graphics device capabilities, characteristics, and states; controlling graphics devices; generating and controlling groups of primitives called segments; and obtaining graphical input. At present, the CGI supports only 2D output primitives and controls only one device. The CGI has been designed to directly support GKS viewing operations.

The purpose of the CGI is to serve as a standardized, device independent interface for graphics package implementers to write to. When supported in hardware by peripheral device manufacturers, the burden of writing device drivers will be greatly eased. Furthermore, just as with applications written to GKS, implementations written to the CGI will be able to take advantage of new hardware without having to be rewritten and extensively modified. Thus, the developer's investment is protected, and any application layered on top of the CGI shares this same benefit. With hardware products having a lifespan of barely one year (new, less costly, faster, and higher-resolution devices usually appear within a year of the initial product offering), writing to the CGI instead of to the hardware pays enormous dividends to the developer and end-user (consumer) alike.

1.4 Metafiles

There are two phases to the use of metafiles. To create the metafile, a metafile "device driver," or metafile writer or generator, must be available with a graphics package. To read and redisplay metafiles generated on other computers, a metafile reader, or interpreter, must be available on the system where the picture is to be used.

Two principal kinds of graphical metafiles are being standardized. The GKS metafile (GKSM) is an audit trail of the GKS commands that were used to generate a particular picture on a GKS workstation. Although GKS metafiles may be interpreted

outside the GKS environment, most users will use GKS to both generate and interpret these files. GKS metafiles are not very compact and, if the picture stored in the metafile were designed by the user in an incremental and iterative manner, the metafile may contain a large quantity of superfluous information.

CGM. The Computer Graphics Metafile represents a snapshot of the final image that a program has created. Unlike the GKS metafile, no intermediate pictures are represented in the CGM. Consequently, the files are more compact.

The CGM provides a file format suitable for the storage and retrieval of picture description information. The file format consists of an ordered set of elements that can be used to describe pictures in a completely device-independent way. One or more pictures can be stored in a single metafile, and the metafile is defined in such a way that, in addition to sequential access to the whole metafile, random access to individual pictures is well defined. That is, the pictures are completely independent, one from another: their appearance does not depend upon the order in which they are accessed or displayed.

In addition to a functional specification, the CGM standard documents three standard encodings of the metafile semantics. The Character encoding requires minimum metafile size and is suitable for transmission across networks of heterogeneous systems but is expensive to encode and decode. The Binary encoding requires minimum effort to generate and interpret but is not well suited for exchange between computers of different arithmetic data types. It is nearly as efficiently coded as the Character encoding. The Clear-text encoding provides maximum readability and editability for ease of use by humans (e.g., for debugging purposes) but, generally, pays a heavy penalty in size and performance. The size is much larger because English and other natural languages contain a lot of redundancy. The performance is worse because parsing and recognizing text strings and converting text strings to internal numbers for use by a graphics subsystem is expensive in its use of CPU cycles.

The standardized CGM elements by type can be found in Table 1. The ESCAPE and APPLICATION DATA elements have been provided to support uses of the CGM in ways that go beyond the exchange of pictures. Nongraphical data and graphical elements not yet standardized can be incorporated into CGMs in a regular way. When these extended metafiles are exchanged by cooperating processes, standard commercial products can be used to handle the standard metafile elements, and new code need be written only for the special, non-standardized elements. Large groups of users of extended metafiles can get together and agree upon a set of extensions--just like MAP and TOP users have agreed upon guidelines to the implementation of the OSI standards. For

example, the elements of a business chart--like legend entries, tick marks, and axis labels--or the elements of a project schedule--like PERT chart symbols, milestone markers, or title--could be marked in the metafile. An editing program could be written to read such metafiles and allow modifications to them before rendering the chart on a hardcopy device or including it in a report or manual.

TABLE 1. CGM ELEMENTS

Element Type	Elements	
Delimeter	BEGIN METAFILE	END METAFILE
	BEGIN PICTURE	END PICTURE
	BEGIN PICTURE BODY	
Metafile Descriptor	METAFILE VERSION	METAFILE DESCRIPTION
	VDC TYPE	MAXIMUM COLOUR INDEX
	INTEGER/REAL PRECISION	INDEX PRECISION
	COLOUR INDEX PRECISION	COLOUR PRECISION
	METAFILE ELEMENT LIST	DEFAULTS REPLACEMENT
	FONT LIST	CHARACTER SET LIST
	CHARACTER CODING ANNOUNCER	
Picture Descriptor	COLOUR SELECTION MODE	SCALING MODE
	LINE WIDTH SPECIFICATION MODE	
	MARKER SIZE SPECIFICATION MODE	
	PERIMETER WIDTH SPECIFICATION MODE	
	VDC EXTENT	
Control	VDC INTEGER/REAL PRECISION	AUXILIARY COLOUR
	CLIP RECTANGLE	CLIP INDICATOR
Graphical Primitive	POLYLINE	DISJOINT POLYLINE
	POLYMARKER	TEXT
	RESTRICTED TEXT	APPEND TEXT
	POLYGON	POLYGON SET
	CELL ARRAY	GDP
	CIRCLE	CIRCULAR ARC
	CIRCULAR ARC CLOSE	ELLIPSE
	ELLIPTICAL ARC	ELLIPTICAL ARC CLOSE

TABLE 1. CGM ELEMENTS (Continued)

Element Type	Elements	
Attribute	LINE BUNDLE INDEX	LINE TYPE
	LINE WIDTH	LINE COLOUR
	MARKER BUNDLE INDEX	MARKER TYPE
	MARKER SIZE	MARKER COLOUR
	TEXT BUNDLE INDEX	TEXT FONT INDEX
	TEXT PRECISION	TEXT COLOUR
	CHARACTER EXPANSION FACTOR	CHARACTER SPACING
	CHARACTER HEIGHT	CHARACTER ORIENTATION
	TEXT PATH	CHARACTER SET INDEX
	ALTERNATE CHARACTER SET INDEX	FILL BUNDLE INDEX
	INTERIOR STYLE	FILL COLOUR
	HATCH INDEX	PATTERN INDEX
	PERIMETER TYPE	PERIMETER WIDTH
	PERIMETER COLOUR	PERIMETER VISIBILITY
	FILL REFERENCE POINT	PATTERN SIZE
	PATTERN TABLE	COLOUR TABLE
	ASPECT SOURCE FLAGS	
Escape	ESCAPE	
External	APPLICATION DATA	MESSAGE

1.5 Purposes of Metafiles

Figure 3 shows three levels of data interchange. At the lowest level, simple files can be exchanged between systems (path O). The file transfer, access, and management (FTAM) standard--part of the OSI level 7 facilities--is designed to handle this activity, but knows nothing about the semantics of the contents of the files it transfers. In its full generality, it will automatically convert the format (i.e., syntax) of files as part of the transfer process if the file types have been registered. Indeed, the three encodings of the metafile are being registered for use with FTAM. Although FTAM knows nothing about graphics, FTAM can be used to transfer graphical metafiles from system to system (paths I and J).

At the next level of exchange, graphical metafiles are used to transfer pictures, drawings, or images between graphical processes (path G). In this case, not only is the syntax known to the cooperating processes, but also the semantics; that is, both processes know about color tables, bundled attributes, filled areas, and pixel arrays. However, they don't know about any application-specific information like surfaces and centers of mass, nor relationships between entities like the association between two connected objects. Only extended metafiles that use APPLICATION DATA elements are able to communicate information that goes beyond the elementary picture representation information. CGM is the principal graphics standard at this level. Picture files may be sent directly to print spooler tasks (path K) and thence to hardcopy devices (path N), or they may first be manipulated by application-specific programs like desktop publishing applications (path H). Such programs, perhaps written using GKS,will manipulate the pictures, merge the images with other, non-graphical information such as text and document layout commands from a word processing program, and then route the layed-out pages to a print spooler (path L) or directly to a hardcopy device like a laser printer (path M). The spooler and other application programs are likely to use CGI to talk to the graphics hardcopy devices. These application programs may also produce structured, editable output like full drawings that may again be treated as product databases (path F).

At the highest level of exchange, one needs to transfer product databases. In the CAD/CAM/CIM environment, the product definitions often have a high degree of geometric information; indeed, if the product is a drawing, it may consist principally of information that is pictorial. However, the purpose of a product database is to support such functions as design, analysis, manufacturing and testing. Furthermore, it is sometimes desirable that cooperating processes not only share a product database, but also that they share views of the same objects so that, for example, engineers may see the same view and consult with each other concerning the object's design and

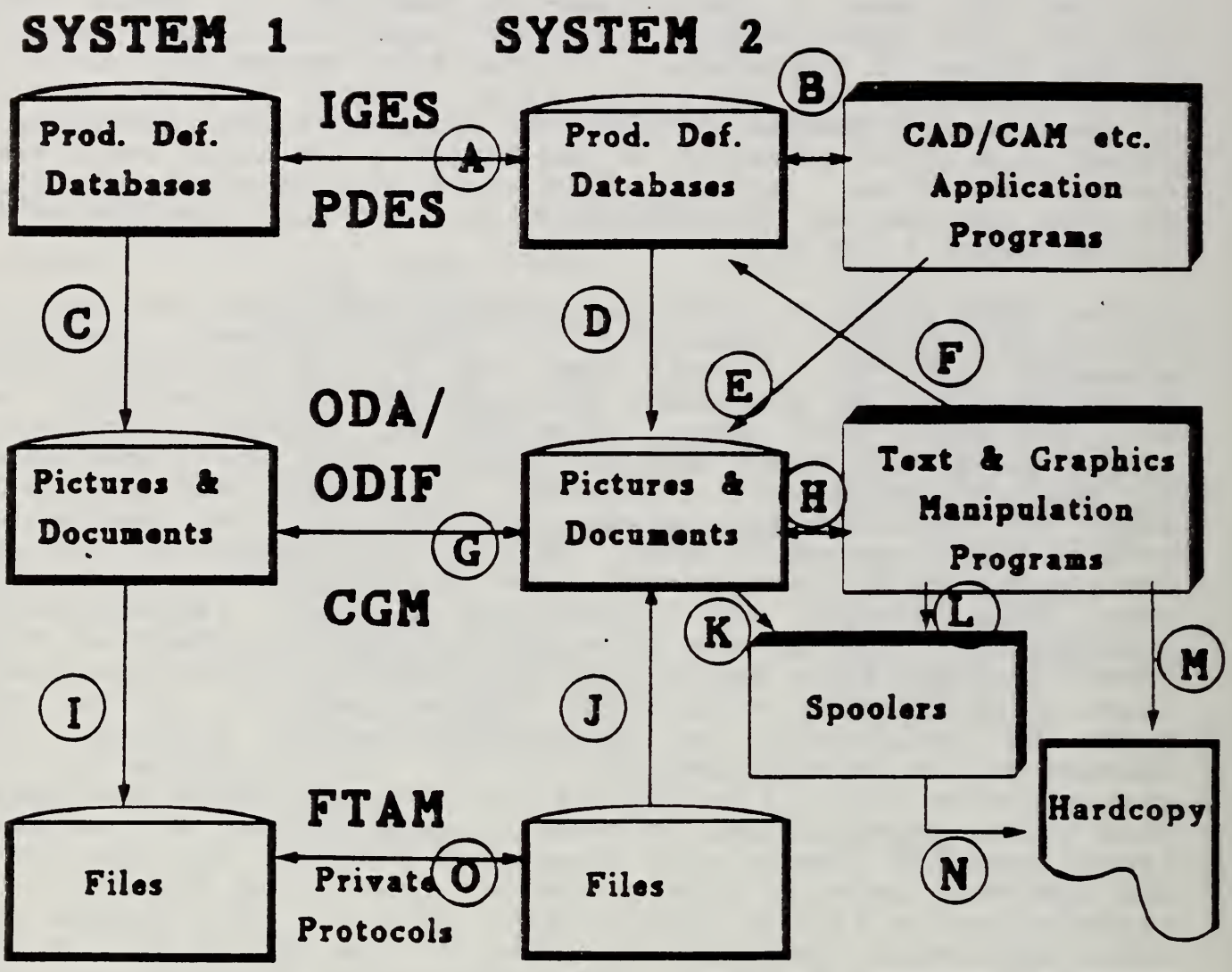

SYSTEM 1
SYSTEM 2
Prod. Def. Databases
IGES
PDES
A
Prod. Def. Databases
B
CAD/CAM etc. Application Programs
C
D
E
F
ODA/
ODIF
Pictures & Documents
G
Pictures & Documents
H
Text & Graphics Manipulation Programs
CGM
K
L
M
Spoolers
I
J
FTAM
Hardcopy
Files
Private
Protocols
O
Files
N

Figure 3.

manufacture. The two principal standards at this level (path A) are the Initial Graphics Exchange Specification (IGES) and the Product Definition Exchange Specification (PDES). Of particular interest are the PDES Presentation Entities. These two standards are described in more detail in section 1.7 below.

CGM pictures can be derived directly from IGES or PDES formatted databases (paths C and D) for archiving and for inclusion in technical manuals and reports. Typical CAD/CAM programs, again perhaps written using PHIGS to obtain device independence, will build internal geometric models and data structures by loading data from an external product definition database (path B). During the processing, CGM picture files may be produced (path E) for later use in such applications as desktop publishingand picture previewing.

1.6 Picture File Transfer and Storage

A graphical metafile, which is used to transfer and store pictures, has great value in a networked environment. For example, PCs are being connected to large mainframe hosts for number-crunching and access to the corporate or engineering databases. Because the users are trying to exchange information (and not just data) and because pictures convey information much more efficiently than words and numbers, graphical metafiles, especially those extended to handle text and graphics, will play an important role in integrated environments. Because graphical metafiles store pictures in a resolution independent manner, these pictures can be previewed on low-cost displays like those found on today's PCs and still be printed on high-resolution printers, plotters, and camera systems. Metafiles can also be stored for later use--in fact, can be stored for years and then be retrieved for plotting on devices with new capabilities and resolutions not imagined when the metafile was created.

In the text and office systems arena, at this level of interchange, the Office Document Architecture/Office Document Interchange Format (ODA/ODIF) standard (ISO DIS 8613) is also found. It is used to exchange so-called "compound documents" that may contain a mixture of pure text, graphics pictures, and facsimile images. The current draft proposal for a Computer Graphics Content Architecture contains the CGM as its base technology. Thus, the CGM is a very important standard for such applications as technical documentation, maintenance manuals, and project plans.

1.7 Product Definition Database Transfer and Storage

The term <u>product data</u> denotes the totality of data elements that completely define the product for all applications over its

expected life cycle. Product data includes the geometry, topology, relationships, tolerances, attributes, and features necessary to completely define a component part or an assembly of parts for the purposes of design, analysis, manufacture, test and inspection.

IGES. The Initial Graphics Exchange Specification is a mature mechanism for the digital exchange of database information among present-day CAD systems. Now in its third version, engineering drawings, 3D wireframe and surfaced part models, printed wiring product descriptions, finite element mesh descriptions, and process instrumentation diagrams are addressed by IGES. IGES information, including drawings and 3D wireframe product models, is intended for human interpretation at the receiving site.

PDES. Whereas IGES has addressed the need for data exchange where the received product model is interpreted by a human either as a display or as a generated plot, the Product Data Exchange Specification (PDES) project is focussed on exchanging product models with sufficient information content as to be interpretable directly by advanced CAD/CAM application programs. In addition to geometry, PDES will support a wide range of non-geometry data such as manufacturing features, tolerance specifications, material properties, and surface finish specifications.

It must be recognized that IGES and PDES have different technological objectives and are in vastly different stages of development. IGES is mature and in production; PDES is in its early stages of development and will not be ready for use before the late 1980's. IGES is a US standard, while PDES represents the US contribution to the international effort in product data exchange (ISO TC184). Until PDES has been proven, IGES will continue to evolve with upward compatible versions to support the commitments already made by industry. Much of the development work on PDES is expected to benefit this continuing IGES work.

Although much of PDES version 1.0 will have little to do directly with graphics, the early work on PDES includes a task to develop a conceptual schema to support the mechanical design of flat plates with circular holes. Wireframe geometry will be used. The schema will support some user-view presentation (viewing) scenarios pertinent to this area of mechanical design. Wireframe geometry entities and the presentation entities have been developed as part of this task. These entities use concepts drawn from the ASC X3H3 graphics standards.

1.8 Transaction Recording

Sometimes there is a need to record everything that is passed across an interface. Such transaction files are called audit trails. Audit trails can be used for a variety of purposes.

Audit trails provide a simple graphical metafile for the exchange of pictures. The GKS metafile (GKSM) is an example of such a metafile. The GKSM stores all the output transactions at the GKS workstation interface. Consequently, segment contents and segment manipulations are recorded, as well as the usual output primitive and attribute information similar to that recorded in the CGM. Audit trails like the GKSM typically are useful only within a homogeneous graphics system environment, because they are expensive to interpret unless the underlying graphics system is also GKS. Audit trails are also not very compact, because superfluous data may be placed into the metafile--data that records intermediate pictures between those pictures of value to the program.

Audit trails are particularly well suited as graphical restart/recovery mechanisms. The GKSM can be activated concurrently with, say, the graphics display screen. As commands are issued to the GKS workstations to cause picture elements to appear to the operator, they are also captured in the GKSM. If the program were to be aborted unexpectedly, the GKSM could be interpreted at the beginning of the next session to cause the previous picture to be recreated and the GKS system to be placed into the same state that it was in when the program terminated abnormally.

1.9 Symbol Library

Many graphical applications, especially in the CAD/CAM area, need to start with a collection of graphical symbols that can be used by the operator to build up more complex pictures and drawings. Rather than have the application create these standard symbols from scratch each time a new session starts, it is faster and more convenient to be able to read in the symbol definitions from some external source. Depending upon how these symbols are to be used by the application, a variety of metafiles can be used for this purpose.

In general, the metafile used is a function of the kind of graphical manipulations needed by the application or allowed by the underlying graphics support system. If the system is GKS, a CGM or a GKSM can be used to store symbol instances used to load a symbol library. Each CGM picture or each GKSM segment can correspond to a symbol that can be stored in the Workstation Independent Segment Storage of GKS.

If the underlying system is PHIGS, the application is probably using the PHIGS centralized Structure Store for both viewing and modelling of the graphical elements. The PHIGS Archive File mechanism has been designed to allow the saving and restoring of complete structure networks from the Structure Store. PHIGS

defines a complex set of rules for resolving what happens if structures are saved into Archive Files when the networks are not completely defined, and what happens if structures are retrieved from Archive Files when the Structure Store already contains a structure of the same name.

If the application is dealing with product databases, it may be necessary to load an entire database prior to allowing the operator to start interacting with the application. But IGES and PDES files can be very large, with complex interrelationships specified within the file; consequently, they are often unsuitable as symbol libraries which would be loaded each time the program is run. Instead, non-standardized, proprietary product databases may be linked into the application when it is initialized.

In sum, any product or graphical picture file format can serve as a symbol library. The appropriate file format to use depends on the application and the underlying graphical system on which the application is built. For portability of symbol libraries, the PHIGS Archive File and CGM offer the best solution, while maintaining good performance and size characteristics.

2. NBS ROLE IN COMPUTER GRAPHICS STANDARDS

NBS's role in graphics standards encompasses the following areas:

- o Active participation in voluntary standards committees
- o Promulgation of FIPSs (Federal Information Processing Standards)
- o Development of test methods for graphics standards
- o Registration authority for the ISO

2.1 Participation in Voluntary Standards Groups

NBS has been a member of the Accredited Standards Committee X3H3-which is responsible for computer graphics standards-since X3H3's inception. Since the federal government is the largest single user of computer software, NBS's objective is to represent the interests of federal agencies to the committee. Since X3H3's membership is primarily computer graphics vendors, this representation also serves to help balance the representation between vendors and users.

In the past, NBS's technical contributions to X3H3 were primarily in language binding. An NBS representative chaired the X3H3 Language Binding Task Group, and initiated the bindings of the Graphical Kernel System to Fortran, Ada, C, Basic, and Pascal. An NBS representative currently serves on the Formal Description, Validation and Testing Task Group within X3H3 and its counterpart within ISO TC97/SC21/WG2 to pursue interests in graphics standards validation. An alternate X3H3 member from NBS participates in the PHIGS Task Group. PHIGS is of great interest to NBS because it is intended for use with CAD/CAM applications, and is thus of great importance to DOD, NASA, and many other federal agencies.

2.2 Promulgation of FIPS

As needed standards emerge from committees, the process of adopting them as FIPSs begins. GKS was adopted as a FIPS in April 1986. Before a standard becomes a FIPS, its effect on federal agencies is thoroughly analyzed. Furthermore, because of its limited staff, ICST must concentrate on computer standards that have the largest impact. Thus, only the most beneficial voluntary computer standards actually become FIPSs. FIPS GKS is the first FIPS for computer graphics and reinforces the key role computer graphics has in federal operations.

Proposed FIPS have a public review period similar to the ANSI public review period, in which state and federal government agencies comment on the proposed FIPS. Currently, CGM has just

completed this public review cycle in the FIPS process, and is expected to become a FIPS in late 1986.

2.3 Test Suite Evaluation

In 1982 the European Economic Community (now called Commission of the European Communities, or EC) sponsored a workshop in Rixensart, Belgium, to bring together all parties interested in contributing to the development of routines to test for conformance to ISO GKS. The design and coding for some of these routines began shortly thereafter. Most of the actual programming was done at Leicester University in the UK and the Technical University in Darmstadt, West Germany. The US, France, and other countries attended subsequent workshops to review these efforts.

The test suite developed in Europe contains several limitations. Tests have been developed only at the application and operator interfaces. Only selected levels of the standard have been tested, and the tests that exist are written in Fortran and test only for Fortran bindings to the standard. Additionally, no test plan was developed to evaluate the accuracy of the suite.

At present NBS has acquired the European GKS test suite from Gesellschaft fur Mathematik und Datenverarbeitung MBH (GMD), a research institute in West Germany that is distributing the suite within Europe. An agreement between NBS and GMD authorizes NBS to distribute the test suite to companies in the US and Canada. Participating companies provide feedback to NBS concerning the quality and utility of the tests. Currently, 30 companies have signed agreements with ICST to participate in evaluating the test suite.

At the international level, the EC is funding an effort to develop a harmonized graphics standards test suite and testing service throughout Europe. Organizations in the UK, West Germany, and France are the subcontractors for the EC in this effort. The US is participating in this effort through the ISO TC97/SC21/WG2 Validation and Testing Rapporteur Group, and will forward the US companies' evaluation of the test suite to the EC.

2.4 Registration

All proposed and anticpated standards emerging from ANSI X3H3 and its international counterpart ISO TC97/SC21/WG2 share certain classes of graphical items that are allowed to vary across implementations of the standard. Line type is an example of such a class. Four mandatory line types are required in GKS and CGM, and the values 1,2,3, and 4 are reserved for them; line-type values 5 or greater are implementation defined and are reserved

for registration. The other classes of graphical items to be registered are marker type, hatch style, text font usage, prompt echo type, error message, escape, and GDP (graphics drawing primitive). The purpose of registering these implementation-defined graphical items is to encourage implementors using the same graphical items to reference them in the same way. The purpose of the register is to inform all concerned of items already registered, and of the specific identifiers assigned to them.

NBS was selected by ISO as the registration authority for graphical items. NBS's responsibility is to maintain a register of the names and meanings assigned to these graphical items. NBS will receive proposals from various sponsoring bodies for meanings to be assigned to graphical items. It will coordinate the acceptance or rejection of these proposals with ISO TC97/SC21 and assign identifiers to the proposals accepted. NBS will then make the information in this register available to all interested parties.

3. STATUS OF GRAPHICS STANDARDS

3.1 At the International Level

NBS participated in the meeting of the ISO TC97/SC21/WG2 Formal Description Validation and Testing Rapporteur Group (FDVT Rap Group) held March 8-16, 1986 in the U.K. The European Community has subcontracted efforts to convert GKS Level 2B tests to the programming language C (effort to take 12 months); development of CGM tests (effort to take 18 months); and development of GKS-3D tests (effort to take 24 months). The progress of this work will be monitored closely. In addition, there are missing pieces in this work which are not funded at the present time. In particular, test routines for PHIGS and CGI, and the conversion of GKS tests to Pascal and Ada are not being pursued.

NBS is trying to ensure that one coordinated set of graphics test routines is developed for all graphics standards. As part of this effort, NBS is currently distributing the test routines for GKS received from West Germany to companies in North America, asking for feedback concerning the quality and utility of the tests from the implementations.

Members of this International committee have initiated a plan for a series of international workshops on 'Applications of Graphics Standards.' The plan calls for these workshops to be co-sponsored by NBS and Eurographics (and perhaps other European organizations). The first workshop is tentatively planned to be held at NBS in September 1987. Future workshops are planned at yearly intervals, concentrating on CGM and GKS-3D, rotated among the U.S.,the U.K., and West Germany. These conferences will concentrate on scrutinizing implementations of the graphics standards.

The current status of the graphics standards in the international arena is as follows:

Standard	Status	Document
GKS	International Standard	ISO 7942
GKS-FORTRAN	DIS Text produced	DIS 8651/1
GKS-PASCAL	DIS Text almost produced	DIS 8651/2
GKS-Ada	2nd DP ballot closes Apr 16	DP 8651/3
GKS-C	Circulate 2nd WD soon	WD
	Register as DP in Sept 86	(SC21/N669)
GKS-3D	DIS Text produced by July 86	
GKS-3D-FORTRAN	Circulate now for DP ballot	WD
	Register as DP in Sept 86	(SC21/N620)
GKS-3D-PASCAL	Not started yet	---------
PHIGS	ISO WD	
PHIGS-FORTRAN	Circulate 2nd WD soon	WD
	Register as DP in Sept 86	(SC21/N667)

Standard	Status	Document
PHIGS-Ada	Circulate as DP soon Register as DP in Sept 86	WD (SC21/N668)
CGM	Published mid 87	DIS 8632/1-4
CGI	ISO WD	

Note: There are four formal development stages within ISO:
WD = Working Draft (1st)
DP = Draft Proposed Int'l
Standard (2nd)
DIS = Draft Int'l Standard (3rd)
IS = Int'l Standard (4th)

3.2 At the National Level

The current status of the national graphics standards efforts is as follows:

Standard	Status	Document
GKS	National standard	ANSIX3.124 1985
GKS-FORTRAN	National standard	ANSIX3.124.1 1985
GKS-PASCAL		dpANSIX3.124.2
GKS-Ada		dpANSIX3.124.3
GKS-C		----------
GKS-3D	Begin processing as dpANS June 86	
GKS-3D-FORTRAN		----------
PHIGS	2nd Public review Fall 86	dpANSIX3.144
PHIGS-FORTRAN	Recommended as dp by X3H3 1st public review this summer	X3H3/85-63
PHIGS-Ada		X3H3/86-43
PHIGS-C	Working draft	X3H34/85-82
CGM	Published as ANS this summer	dpANSIX3.122
CGI	X3H3 Letter Ballot on ISO WD	

3.3 At the Federal Level

The status of graphics standards in the FIPS (Federal Information Processing Standards) arena is as follows:

Standard	Status	Document
GKS	FIPS	FIPS 120
CGM	Public Review over Secretary package in Preparation	

APPENDIX 2

FEATURE COMPARISONS AMONG GRAPHICS STANDARDS

APPENDIX 2

FEATURE COMPARISONS AMONG GRAPHICS STANDARDS

Generic Functions	CGI	CGM	GKS	GKSM	PHIGS	ARCH
Graphics System Management Functions						
Open	-	-	*	-	*	-
Close	-	-	*	-	*	-
Activate Workstation	-	-	*	-		-
Deactivate Workstation	-	-	*	-		-
Virtual Device Management Functions						
Initialize	*	-	*	-	*	-
Reset to Defaults	*	-		-		-
Terminate	*	-	*	-	*	-
Make Picture Current	*	-		-		-
Set Deferral Mode	o	-	*	*	*	-
Clear View Surface	*	-	*	*		-
Background Colour	o	*	!	!	!	
File Delimiter and Descriptor Elements						
Begin File	-	*	-	*	-	*
End File	-	*	-	*	-	*
Begin Picture	-	*	-		-	*
Begin Picture Body	-	*	-		-	
End Picture	-	*	-		-	*
File Version	-	*	-	*	-	*

Generic Functions	CGI	CGM	GKS	GKSM	PHIGS	ARCH
File Description	-	*	-	*	-	*
File Elements List	-	*	-		-	
File Defaults Replacement	-	*	-		-	
Metafile/Archive File Functions						
Open Archive File		-		-	*	-
Close Archive File		-		-	*	-
Write Item to Metafile		-	*	-	!	-
Archive Structure Networks		-		-	*	-
Archive All Structures		-		-	*	-
Set Conflict Resolution		-		-	*	-
Get Item from Metafile		-	*	-	-	-
Retrieve Structure Networks		-		-	*	-
Retrieve All Structures		-		-	*	-
Read Item from Metafile		-	*	-		-
Interpret Item		-	*	-	!	-
Delete Structures from Archive		-		-	*	-
Delete Structure Networks from Archive		-		-	*	-
Delete All Structures from Archive		-		-	*	-
Coordinate Space Control Functions						
Window			*			
Viewport			*	*		
Normalization Transformation			*			
VDC Type	o	*				
VDC Precision for Integer Points		o	*			
VDC Precision for Real Points	o	*		!		

Generic Functions	CGI	CGM	GKS	GKSM	PHIGS	ARCH
VDC Extent	*	*				
Device Viewport	*		*	*	*	
Device Viewport Specification Units		o				
Device Viewport Specification Mapping		o				
Clip Rectangle	*	*	*	*	*	
Clip Indicator	*	*	*	!		
Error Functions						
Pop Error Stack	o	-		-		-
Empty Error Stack	o	-		-		-
Emergency Close System		-	*	-	*	-
Error Handling		-	*	-	*	-
Error Logging		-	*	-	*	-
Error Handling Mode		-		-	*	-
Miscellaneous Control Functions						
Integer Precision	o	*		!		
Real Precision	o	*		!		
Index Precision	o	*		!		
Colour Precision	o	*		!		
Colour Index Precision	o	*		!		
Maximum Colour Index	*	*				
Character Coding Announcer	o	*				
Message	o	*	*	*	*	
Application Data	o	*		*	*	*
Escape	*	*	*	*	*	*
Output Functions						

Generic Functions	CGI	CGM	GKS	GKSM	PHIGS	ARCH
Polyline	*	*	*	*	*	*
Disjoint Polyline	*	*				
Circular Arc 3 Point	o	*				
Circular Arc Center	*	*				
Elliptical Arc	o	*				
Polymarker	*	*	*	*	*	*
Text	*	*	*	*	*	*
Restricted Text	*	*				
Append Text	*	*				
Polygon	*	*	*	*	*	*
Polygon Set	o	*	3	3	*	*
Rectangle	*	*				
Circle	*	*				
Circular Arc 3 Point Close	o	*				
Circular Arc Center Close	*	*				
Ellipse	*	*				
Elliptical Arc Close	o	*				
Cell Array	*	*	*	*	*	*
GDP	o	*	*	*	*	*
Attribute Functions						
Line Bundle Index	*	*	*	*	*	*
Line Type	*	*	*	*	*	*
Line Width	o	*	!	!	*	*
Line Colour	*	*	*	*	*	*
Line Representation	o		*	*	*	

Generic Functions	CGI	CGM	GKS	GKSM	PHIGS	ARCH
Marker Bundle Index	*	*	*	*	*	*
Marker Type	*	*	*	*	*	*
Marker Size	o	*	!	!	*	*
Marker Colour	*	*	*	*	*	*
Marker Representation	o		*	*	*	
Text Bundle Index	*	*	*	*	*	*
Text Font Index	o	*	!	!	*	*
Text Precision	*	*	!	!	*	*
Character Expansion Factor	o	*	*	*	*	*
Character Spacing	o	*	*	*	*	*
Text Colour	*	*	*	*	*	*
Character Height	*	*	*	*	*	*
Character Orientation	o	*	!	!	*	*
Text Path	o	*	*	*	*	*
Text Alignment	*	*	!	*	*	*
Character Set Index	o	*				
Alternate Character Set Index	o	*				
Text Representation	o		*	*	*	
Fill Bundle Index	*	*	*	*	*	*
Interior Style	o	*	*	*	*	*
Fill Colour	o	*	*	*	*	*
Hatch Index	o	*	!	!	*	*
Pattern Index	o	*	!	!	*	*
Fill Reference Point	o	*	*	*	*	*
Pattern Plane	-	-	*	*	*	*

Generic Functions		CGI	CGM	GKS	GKSM	PHIGS	ARCH
Pattern Table		o	*	*	*	*	
Pattern Size		o	*	!	*	*	*
Fill Representation		o		*	*	*	
Edge Bundle Index		*	*	3	3	*	*
Edge Type	*	*	3	3	*	*	
Edge Width		o	*	3!	3!	*	*
Edge Colour		*	*	3	3	*	*
Edge Visibility		o	*	3	3	*	*
Edge Representation		o		3	3	*	
Add Names to Set						*	*
Remove Names from Set						*	*
Output and Attribute Control Functions							
Scaling Mode			*				
Colour Selection Mode		o	*				
Colour Value Extent		o	*				
Auxiliary Colour		o	*				
Transparency		*	*				
Colour Table		o	*	!	!	!	
Line Width Specification Mode		o	*				
Edge Width Specification Mode		o	*				
Marker Size Specification Mode		o	*				
Aspect Source Flags		*	*	*	*	!	*
Font List		o	*				
Character Set List		o	*				
Begin Figure		o					

Generic Functions	CGI	CGM	GKS	GKSM	PHIGS	ARCH
End Figure	o					
Close Partial Figure	o					
Implicit Edge Visibility	o					
Set Colour Model					*	
Modelling Transformations						
Local Transformation				*	!	
Global Transformation					*	*
Translate				*		
Scale					*	
Rotate X, Y, or Z					*	
Rotate					*	
Define Coordinate System					*	
Compose Matrix					*	
Transform Point					*	
Viewing Functions						
View Index			3	3	*	*
View Representation			3	3	!	
View Matrix					*	
View Mapping					*	
View Reference Point					*	
View Plane Normal					*	
View Up					*	
HLHSR Identifier			3	3		
HLHSR Mode			3	3		
Evaluate View Matrix		-	3	-	*	

Generic Functions	CGI	CGM	GKS	GKSM	PHIGS	ARCH
Evaluate Transformation Matrix		-	*	-	*	
Accumulate Transformation Matrix	-	*	-	*		
Segment/Structure Manipulation Functions						
Request Segment Identifier	o					
Open Segment	*		*	*	!	
Close Segment	*		*	*	!	
Copy Segment	o		!		*	
Delete Element/Range/Between Labels					*	
Delete Segment	*		*	*	*	
Delete All Segments	o		!		*	
Empty Structure					*	
Execute Structure					*	*
Label					*	*
Set/Offset Element Pointer (at Label)					*	
Rename Segment	*		*	*	!	
Change Structure (Identifier and) References				*		
Redraw Segment (Post Root)	*				!	
Redraw All Segments	*		*	*	!	
Update	o		*	*	!	
Implicit Segment Regeneration Mode	*		!	!	!	
Copy Segment to Workstation			*			
Associate Segment with Workstation			*		!	
Insert Segment			*		!	
Segment Attribute Functions						
Segment Transform	*		*	*	!	

Generic Functions	CGI	CGM	GKS	GKSM	PHIGS	ARCH
Segment Visibility	*		*	*	!	
Segment Highlighting	*		*	*	!	
Segment Display Priority	*			*		
Raster Functions						
Pixel Array	*					
Mapped Bitmap Foreground Colour	o					
Create Bitmap	*					
Delete Bitmap	*					
Select Drawing Bitmap	*					
Display Bitmap	*					
Two Operand Bitblt	*					
Tile Two Operand Bitblt	*					
Tile Three Operand Bitblt	o					
Drawing Mode	*					

KEY TO ABBREVIATIONS AND SYMBOLS

COLUMN HEADINGS:
- GKS = Graphical Kernel System
- GKSM = GKS Metafile
- CGI = Computer Graphics Interface
- CGM = Computer Graphics Metafile
- PHIGS = Programmers' Hierarchical Interactive Graphics System
- ARCH = PHIGS Archive File

SYMBOLS:
- * = required in the standard
- o = optional in the standard
- ! = not exactly the same as in CGI (if GKS) or as in GKS (if PHIGS)
- - = inappropriate function for that standard
- 3 = in GKS-3D only (not in GKS)

APPENDIX 3

IGES ENTITIES AS RENDERED BY GRAPHICS STANDARDS SYSTEM

APPENDIX 3

IGES ENTITIES AS RENDERED BY GRAPHICS STANDARDS SYSTEM

This Appendix describes each of the IGES entities in turn, noting how well these entities could be rendered by a system based upon the graphics standards.

1. Geometric Entities

CIRCULAR ARC. Not available in GKS/PHIGS. Present as CIRCLE and 3-POINT ARC in CGI/CGM.

COMPOSITE CURVE. Rendered by POLYLINE in GKS/PHIGS; by POLYLINE, ARC, and ELLIPTICAL ARC in CGI/CGM. The full conics and splines of IGES are not directly supported, but would have to be approximated by POLYLINE.

CONIC ARC. Not available in GKS/PHIGS. Present as CIRCLE, ELLIPSE, 3-POINT ARC, and ELLIPTICAL ARC in CGI/CGM. Parabolas and hyperbolas are not directly supported.

COPIUS DATA. Directly available as POLYLINE with different line types.

PLANE. Unbounded planes need no direct visualization. Bounded planes correspond to POLYGON SET in GKS/PHIGS/CGM and are supplemented by CLOSED FIGURE in CGI.

LINE. Available as a two-point POLYLINE.

PARAMETRIC SPLINE. Can be visualized by lines, arcs, etc. However, information about the shape is lost.

PARAMETRIC SPLINE SURFACE. Can be visualized by rectangles, lines, arcs, etc. However, information about the shape is lost.

POINT. Available as POLYMARKERS for pre-defined symbols and positioned SEGMENTS (GKS/CGI but not CGM) or STRUCTURES (PHIGS) for user-defined symbols.

RULED SURFACE. Can be visualized by lines, etc. However, information about the shape is lost.

SURFACE OF REVOLUTION. Can be visualized by lines, arcs, etc. However, information about the shape is lost.

TABULATED CYLINDER. Can be visualized by lines, arcs, etc. However, information about the shape is lost.

TRANSFORMATION MATRIX. Really used like an attribute in IGES. Also special form numbers are used with certain contructs for Finite Element Analysis and viewing. Closely related to SEGMENT TRANSFORMATIONS of GKS and PHIGS transformation matrix structure elements used for modelling and viewing. No direct parallel in CGI or CGM.

FLASH ENTITY. In GKS, only have POLYMARKER or segments to realize the forms. In CGI/CGM, you also have CIRCLE and RECTANGLE. IGES flash entity form 4, rounded rectangle, is not directly available in any of the graphics standards.

RATIONAL B-SPLINE CURVE. Can be visualized by lines, arcs, etc. However, information about the shape is lost. Parabolas and hyperbolas not directly supported by the graphics standards.

RATIONAL B-SPLINE SURFACE. Can be visualized by lines, arcs, etc. However, information about the shape is lost. Parabolas and hyperbolas not directly supported by the graphics standards.

OFFSET CURVE. Can be visualized by lines, arcs, etc. However, information about the relationship between the two entities is lost.

CONNECT POINT. Can be visualized by lines. However, information about the relationship between the entities is lost.

NODE. Not directly visualized, so no need for support from the graphics standards. However, not unlike a PHIGS structure label.

FINITE ELEMENT. Thirty-three topologies are currently defined by IGES version 3. All can be visualized using the graphics primitives of line, arc, ellipse, etc., with loss of information concerning the relationships between the lines and curves making up the wire-frame model. Furthermore, the IGES specification is much more compact.

NODAL DISPLACEMENT AND ROTATION. These are attributes that are used to communicate finite element post processing data.

OFFSET SURFACE. Can be visualized by lines, arcs, etc. However, information about the relationship between the two entities is lost.

CURVE ON PARAMETRIC SURFACE. Can be visualized by lines, arcs, etc. However, information about the relationship between the curve and the surface is lost.

TRIMMED (PARAMETRIC) SURFACE. Can be visualized by lines, arcs, etc. However, information about the relationships among the boundary lines and other curved lines used to represent the surface is lost.

ANGULAR DIMENSION. Visualize using text, lines, arrows, and arcs. Only arrows not directly supported in the graphics standards. See discussion under LEADER entity below.

CENTERLINE. Use graphics circles (in CGI/CGM) only and special line types. Could be made available in GKS/PHIGS through special marker types.

DIAMETER. Visualize using text, lines, arrows, and arcs.

FLAG NOTE. Visualize using text and fill area (polygon) primitives.

GENERAL LABEL. Visualize using text, lines, and arrows.

GENERAL NOTE. A general note entity consists of one or more text strings. Each text string contains text, a starting point, text size, and angle of rotation of the text. A single font number applies to the whole note and incorporates the separate concepts of type face (appearance of the characters; e.g., bold Helvetica, italic Futura) and character set (shape of the characters; e.g., ASCII, German National Set, Math, Greek). Only 7-bit character codes are supported. In addition, a form number is used to designate the layout of the (possibly multiple) text strings, the justification of the strings within a text rectangle, and whether there are subscripts, superscripts, fractions, and embedded font changes. The GKS/PHIGS TEXT and CGI/CGM TEXT, APPEND TEXT, and RESTRICTED TEXT primitives with their numerous text attributes are capable of visualizing any general note entity. With RESTRICTED TEXT and APPEND TEXT, CGI/CGM are a bit more capable than GKS/PHIGS.

LEADER. Eleven arrow head types are specified in IGES version 3. Although all can easily be rendered using more primitive lines and polgons, the information that the arrowhead belongs with the leader line is lost when described using graphics standards primitives.

LINEAR DIMENSION. Visualized using text, lines, and arrows.

ORDINATE DIMENSION. Visualized using text, lines, and arrows.

POINT DIMENSION. Visualized using text, lines, arrows, circles, and polylines/polygons (hexagons).

RADIUS DIMENSION. Visualized using text, lines, arrows, and arcs.

SECTION. Visualized using FILL AREA with HATCH patterns. Of the eight pre-defined IGES patterns, two correspond to CGI/CGM hatch patterns; namely, IGES form 31 is CGM hatch style 3 and IGES form 37 is CGM hatch style 6.

GENERAL SYMBOL. Visualized using text, lines, arrows, etc., and possibly grouped in SEGMENTS (GKS or CGI but not CGM) or STRUCTURES (PHIGS).

SECTIONED AREA. Visualized with POLYLINE and FILL AREA in GKS/PHIGS/CGM and also using CLOSED FIGURE with fill in CGI. However, the IGES representation is generally more compact, because there is control over the angle and spacing of the lines that make up the cross-hatched fill pattern. In these cases, the DISJOINT POLYLINE element of CGI/CGM may help. In its default state, IGES line pattern code 1 corresponds to CGM hatch style 3, line pattern code 16 to CGM hatch style 6, and line pattern code 18 to CGM hatch style 5.

WITNESS LINE. Visualized using POLYLINES, although the DISJOINT POLYLINE of CGI/CGM may be useful in certain special cases.

ASSOCIATIVITY DEFINITION. Defines the relationship among entities. These relationships are lost in CGM, occasionally can be represented by GKS/CGI segments, and with somewhat greater likelihood can be represented by PHIGS structures.

ASSOCIATIVITY INSTANCE. Many predefined associativity relationships exist in IGES. These include simple and ordered GROUPS (both doubly linked and forward-only linked entities), external references and external reference files, labels, and parent-child structures. These have some parallels in PHIGS structures, but in general need not be directly handled by a graphics viewing system like CGI/GKS. Likewise, the PLANAR and FLOW instance types do not require direct support from the graphics system. The VIEWS VISIBLE and VIEWS VISIBLE, COLOR, LINE WEIGHT instance type have an effect on the visualization of the IGES model. The necessary controls for proper visualization are available in the graphics standards, using viewports, color tables, and line width specification mode. Finally, the DIMENSIONED GEOMETRY instance type has already been discussed in Section 1 above, under the separate elements for ANGULAR DIMENSION, LINEAR DIMENSION, POINT DIMENSION, DIAMETER DIMENSION, RADIUS DIMENSION, and ORDINATE DIMENSION.

DRAWING. A drawing is a collection of annotation entities and views. Multiple drawings can be included in a single file. Drawings have names and size; units may be specified for the drawing. A drawing closely corresponds to a CGM or to any image displayed on a graphics view surface by any of the programmer interface standards: GKS, CGI, or PHIGS.

LINE FONT DEFINITION. Two types of line fonts may be defined. One type considers a line font as a repetition of a basic pattern of visible-blanked segments superimposed upon a straight line or a curve. The other type considers a line font as a repetition of a template figure that is displayed at regularly spaced locations along a planar anchoring curve. None of the graphics standards, at present, support user-defined line types; the type 1 capability would have to be visualized by POLYLINE (in GKS/PHIGS) and DISJOINT POLYLINE (in CGI/CGM); the type two capability by using segments or structures in PHIGS/GKS/CGI and only lines, rectangles, etc. in CGM.

MACRO Capability. This facility allows the IGES specification to be extended beyond the common entity subset, utilizing a formal mechanism which is a part of the IGES Specification. This facility is available in an <u>extremely limited</u> fashion in the graphics standards as ESCAPE and GENERALIZED DRAWING PRIMITIVE (GDP).

PROPERTY. This facility is available in the graphics standards as APPLICATION DATA and MESSAGE. Many engineering specific properties are defined in the IGES specification. A few of them correspond to graphics elements, generally useful for rendering a picture. These are Region Restriction, Hierarchy (partial), Name, Drawing Size and Drawing Units.

SUBFIGURE Definition. All such relationships can be visualized fairly straightforwardly using PHIGS structures, somewhat more difficultly using GKS and CGI segments, and much more difficultly in CGM.

TEXT FONT Definition. This IGES facility provides for the exchange of font definitions, which are limited to a model of the motion of an imaginary pen moving between the points of an integer Cartesian "font coordinate system." None of the graphics standards provide support for "user-defined" fonts. However, such a stroke-precision text capability can easily be built on top of all the graphics standards. Compactness of representation is lost, but speed of picture generation should not be much worse.

VIEW ENTITY. This IGES entity defines a framework for specifying a viewing orientation of an object in three dimensional model space. The framework is also used to support the projection of all or part of model space onto a view plane. One type of projection, an orthographic parallel projection, can be specified. Clipping to a view volume is supported.

The 3-D graphics standards, GKS-3D and PHIGS, have a much richer viewing model. They allow a full 4x4 transformation matrix to be used, thus obtaining perspective projections and various oblique parallel projections as well.

EXTERNAL REFERENCE ENTITY. PHIGS Archive Files would directly support this element. The other standards (GKS/CGI/CGM) would have to do a lot more processing to directly support this feature.

NODAL LOAD/CONSTRAINT ENTITY. A special element to support Finite Element Modelling.

COLOR DEFINITION ENTITY. Not directly available in GKS/PHIGS, but is present in CGI/CGM as the MAXIMUM COLOR EXTENT element. In GKS/PHIGS, the application would query the workstation description tables to gather sufficient information to adjust for this entity before rendering the picture with a graphics system.

TEXT DISPLAY ENTITY. See the earlier discussion under the IGES GENERAL NOTE ENTITY. The attributes of text include height and width, font index, slant angle, rotation angle, mirror flag, and horizontal/vertical text path. All such entities can be correctly displayed by a proper setting of the graphics standards text attributes, which are common to PHIGS/GKS/CGI/CGM.

APPENDIX 4

CORRESPONDENCE BETWEEN PDES ENTITIES AND GRAPHICS ELEMENTS

APPENDIX 4

CORRESPONDENCE BETWEEN PDES ENTITIES AND GRAPHICS ELEMENTS

1. The PDES Viewing Pipeline

By design, the PDES viewing pipeline is modelled after that of PHIGS. All coordinate systems--Local (i.e., Modelling), World, UVN, Normalized Polar Coordinates (NPC), and Device--are three-dimensional, right-handed Cartesian systems. The PHIGS Composite Modelling Transformation is provided by existing IGES entities. The Viewing Transformation is defined by a 4x3 view matrix, whose components may be specified by a view reference point, a view plane normal, and a view up vector. The full view mapping is defined by projection type (PARALLEL or PERSPECTIVE), view plane distance, view plane window, front and back clipping planes and indicators, projection reference point, and NPC viewport. The final Workstation transformation is specified using a Workstation window and a device viewport.

2. PDES Pictures

A PDES picture consists of a viewing operation, a workstation transformation, line, text, and surface attributes, and a so-called presentation list. The presentation list is a linked list of blocks of PDES entities, each block of which can have its own viewing operation, workstation transformation, and line, text, and surface attributes. PDES pictures are analogous to PHIGS structures.

3. PDES Text Model

The PDES Text Model is patterned after the CGI. The three CGI/CGM text entities--TEXT, RESTRICTED_TEXT, and APPEND_TEXT--and the nine text attributes--FONT, PRECISION, EXPANSION_FACTOR, SPACING, COLOR, HEIGHT, ORIENTATION, PATH, and ALIGNMENT--have been included in PDES. Unlike GKS and PHIGS, but like CGI and CGM, the continuous text alignment values are available.

4. PDES Color Model

Unlike any of the graphics standards, the PDES color model permits multiple color tables to be referenced by entities in a single PDES Picture. There is a single, distinguished

BACKGROUND_COLOR associated with each color table. The color table consists of indices associated with RGB triples. The default colors for indices 1 through 8 are black, red, green, blue, yellow, magenta, cyan, and white.

5. PDES Line Attributes

The PDES line attributes--line width scale factor, line type, and color--are nearly identical to those of all the graphics standards. However, in PDES, the line type index is called LINE_FONT and points to a LINE_TABLE where a repeating pattern of bits indicates the desired line type. The default line types for indices 1 through 4 are solid, dashed, dotted, and dash-dotted.

6. PDES Surface Attributes

The PDES surface attributes consist of edge attributes and interior attributes. The edge attributes--edge visibility flag, color, edge type, and edge width scale factor--are taken directly from CGI/CGM with the same generalization to repeating bit patterns as for line type. The interior styles available include HOLLOW, SOLID, PATTERN, HATCH, and EMPTY--exactly those available in CGI/CGM. There is a pattern table and a hatch table. The pattern table includes pointers to the pattern attributes of pattern size, pattern reference point, pattern plane vector, and pattern up vector--attributes equivalent to those in GKS-3D and PHIGS. The hatch attributes are as yet unspecified, but are likely to be such that the default hatch patterns are those of CGI/CGI, with an extension like that of line type to allow the generator of a PDES file to specify exactly the hatch pattern.

APPENDIX 5

STANDARDS FOR SPECIFIC CALS APPLICATIONS

APPENDIX 5

STANDARDS FOR SPECIFIC CALS APPLICATIONS

Project Title: Automated Technical Order System

Goals and Objectives: ATOS is being implemented to improve generation, storage, and distribution time associated with AF TO's. A major goal is to significantly reduce the cost of AF Technical Order acquisition and management.

Contact: Major Edd Higbee

Contact Telephone: (513) 257-3054 AV 787-3054

Contractor(s): Syscon Corp.; San Diego, CA
Rockwell; El Segundo, CA

Acronym: ATOS

Technical Approach: ATOS is an automated publications system for the storage, distribution, revision, and updating of Technical Orders, documents that describe how to operate, maintain and use equipment through narrative text and illustrations. The Phase I Pilot Program installed at the Ogden Air Logistics Center (ALC) consists of central processing units, micro- computer text entry and editing workstations, CAD workstations for creation of illustrations and diagrams, a document image composer and phototypesetter. The system is used to develop change pages to the TO's managed by the Ogden ALC. This pilot system is currently in operation at the Ogden ALC. Final steps have been taken to upgrade the Phase I configuration and replicate the configuration at each ALC and at the Aerospace Guidance and Metrology Center (AEMC).

Phase II introduces contractor interface capability, further enhancement to the individual configurations, and real time electronic distribution of TO's to Technology Repair Centers (TRC's).

Phase III implements electronic distribution to intermediate and organizational maintenance shops at base level and provides the overall system support to meet user needs.

Graphics Standards: CGM can be utilized for changes to pictures within the technical orders. Presently, these changes are made

through the CAD/CAM systems and IGES could then be used to port changes to different hardware (if heterogeneous hardware is used). However, some changes may be cosmetic, and personnel making the changes will probably be graphics artists and not engineers. Thus, it would not be cost effective or efficient to use the CAD/CAM systens for these changes.

IRDS, SQL & NDL: There probably will be a need for an index to the ATOS data. IRDS and/or SQL would be appropriate. Additional information and analysis is needed to determine the extent to which Technical Order and Change Order data is "structured" (vs. lengthy text). NDL or SQL may or may not be appropriate.

Project Title: Engineering Data Computer Assisted Retrieval System

Goals and Objectives: The purpose of EDCARS is to make significant improvements in the quality, retrievability, and cost of Engineering Data.

Contact: Capt. Cauldwell

Contractor: AT&T

Contact Telephone: (513) 257-3054 AV 785-3054

Acronym: EDCARS

Technical Approach: EDCARS is an automated system, similar in concept to ATOS, for capturing, storing, distributing, revising and updating the engineering drawing information currently stored in manual and aperture card form. It is being developed jointly with the Army's Digital Storage and Retrieval Engineering Data System (DSREDS). A phased effort deals with the digitization of selected existing engineering drawings as a prototype system to provide drawing information to selected users. This will then be extended to accept contractor's, CAD/CAM data and new drawings and eventually to provide CAD capability for revision of engineering drawings and specifications. Two-dimensional drawings only.

Graphics Standards: DSREDS/EDCARS is a homogeneous hardware environment. Thus, graphics standards would only be applicable for integration of DSREDS with other systems.

IRDS, SQL & NDL: IBM's IMS database management system is being used to develop a "central index" or directory to the 1.5 million drawings stored on optical disk storage. For compatibility across the services, IRDS (or SQL/NDL) should be planned for future implementations.

Project Title: Integrated Design Support System

Goals and Objectives: The IDS objective is to apply advanced information management technology to integrate engineering data into what will appear to the user to be a "single" data base. IDS, therefore, will mesh with the Integrated Computer Aided Manufacturing (ICAM) program, ATOS, EDCARS, UDB, IMIS and LIMSS.

Contact: T. N. Bernstein

Contractor: Rockwell

Contact Telephone: (513) 255-6992 AV 785-6992

Acronym: IDS

Technical Approach: The IDS program is developing a prototype system to allow the automated storage, retrieval, and transfer of engineering data from design engineering through manufacturing to maintenance and sustaining engineering. The IDS will develop a computer software architecture to manage technical information across the entire life cycle of a weapon systems including data on "lessons learned" which can be fed back to the design function for application to future weapon systems. (See demonstration description in Section D.4.)

Graphics Standards: We haven't been briefed on this system yet. Thus, no information concerning graphics standards is available.

IRDS, SQL & NDL: We haven't been briefed on this system yet. Thus, no information concerning management standards is available. IDS will use some of the IISS technology.

Project Title: Integrated Maintenance Information System Program

Goals and Objectives: To promote effective aircraft maintenance by enhancing capabilities of base level maintenance technicians through tailoring information to support specific needs. IMIS will provide a single interface to all required information and will supplement on-aircraft diagnostics to provide full-fault isolation capabilities.

Contact: Capt. John Von Holle

Contact Telephone: (513) 255-2606 AV 785-2606

Acronym: IMIS

Technical Approach: The IMIS was conceived to integrate the information contained in Air Force ATI systems and to provide access to it at the base level so that maintenance personnel can effectively use it in the performance of their daily activities. An extremely portable, ruggedized computer will provide the technician with a single information source for all of the data he needs to perform his job. The system will display graphic technical instructions, provide intelligent diagnostic advice, provide aircraft battle damage assessment aids, analyze in-flight parameter data, analyze aircraft historical data, and interrogate airborne systems. It will also provide the technician with easy, efficient methods to receive work orders, report maintenance actions, order parts from supply, complete computer-aided training lessons, and transmit messages throughout the maintenance complex. The system will interface with larger computer networks on the base, interface with on-board aircraft systems, or operate independently for dispersed deployments.

The IMIS system will consist of four major subsystems: (1) A technician's portable computer, (2) An aircraft maintenance panel connected to airborne computer/sensors, (3) A maintenance work-station connected to ground-based computer systems, and (4) Integration software which will combine information from the multiple sources and present the data in a consistent way to the technician.

Graphics Standards: Schematics are difficult on a small screen. Concepts inherent in graphics standards, such as windows and viewports would be very useful here.

IRDS, SQL & NDL: The AF is reviewing the content and NDL & SQLformat of the Technical Orders. They plan to have "automatic" cross- referencing/indexing. IRDS or SQL may be appropriate to store the results. Need more information on Technical Order and maintenance data to determine if IRDS or SQL are appropriate.

Project Title: Geometric Modeling Applications Interface

Goals and Objectives: The GMAP program will enable the digital communication of product definition data describing air-cooled jet engine turbine blades and disks throughout the entire product life cycle including engineering analyses, manufacturing, logistics, and depot support. GMAP objectives include: to create a definition and model of engine blade and disk product data; to survey the state-of-the-art geometric modeling systems and application; to define the minimum requirements of a geometric modeling system; and, to demonstrate the integration of a prime contractor with its vendors and Air Force Logistics depots.

Contact: Lt. Robert A. Carringer

Contact Telephone: (513) 255-6976 AV 785-6976

Contractor: Pratt & Whitney

Acronym: GMAP

Technical Approach: To identify blade and disk definitions, Pratt and Whitney will use the walk through technique. At every activity or function in the life cycle of blades and disks, all product description information will be collected. These data will be organized and modeled into information classes of entities. The entities will be installed in the PDDI system.

For data exchange, GMAP will expand the PDDI technology. The PDDI system (exchange format, translator, access software, and conceptual schema) will be updated to incorporate the results of the blade and disk walk throughs. This new version of PDDI will also be upgraded to enable the use of geometric modelers throughout aerospace design, manufacturing, and logistics.

The implementation of GMAP will enable the defense industrial base to eliminate the large quantity of paper drawings that are created by computer aided design (CAD) systems. Air Force acquisition offices and Logistics depots will be able to use the digital data for their automated engineering manufacturing and logistics systems.

Graphics Standards: The emphasis within GMAP and PDDI is on product data vs. graphics data, or CAD/CAM, which emphasizes geometry (line, arcs, circles) vs. CIM which emphasizes features (hole, notch, bend). This seems to eliminate the use of the CGM for GMAP and PDDI unless a product definition database resides behind the CGM. IGES and PDES are what's needed here. However a way of getting graphical info from GKS to PDES would be useful in "feeding" these systems.

IRDS, SQL & NDL: The IRDS is probably appropriate for the Conceptual Schema. Need more information about the "information classes" of entities/features to determine if IRDS, SQL, or NDL are appropriate. (Will use IDEF1* ER model and will need index or catalog database.)

Project Title: Unified Data Base for Acquisition Logistics

Goals and Objectives: The objective of the UDB is to develop, demonstrate, and evaluate a logistics information data base system which will assist weapon system designers and logisticians in incorporating logistics considerations into the early design of weapon systems. The UDB will provide logisticians with a flexible, efficient data base application system designed to ease the burden of documenting iterative LSA tasks.

Contact: Ms. Janet L. Peasant

Contact Telephone: (513) 255-6718 AV 785-3871/6718

Acronym: UDB

Technical Approach: The requirements for an efficient logistics support analysis data base application which also conforms to MIL- STD-1388-2A will be analyzed. A data base management software system will be selected and procured for installation on a government computer. The data base management software will then be installed on an Aeronautical Systems Division (ASD) computer. A data base application will be designed, coded, and tested. Demonstrations using actual and simulated weapon systems logistics data will be conducted to test the quality and efficacy of the software. The demonstrations will be conducted at contractor computer facilities and at government computer sites. The use of dedicated communication lines will be demonstrated. Dial-up communication access will also be demonstrated. Both government and industry logisticians and software engineers will evaluate the initial data base to identify design flaws. The identified faults will be corrected and enhancements such as model interfaces will be identified and implemented. Prior to technology transition, the operational organization will be trained in the use of the system.

System outputs will include all LSA reports in compliance with MIL-STD-1388- 2A including master files required for DoD validations; LSA Control Number Master files, Task Narrative Master file and Parts Master file; Interface to Logistics Analysis models; Work Unit Code List; Support Materials list; Reliability Maintainability Tracking Report and automated output of Contract Data Requirements List (CDRL) items.

The UDB will improve the ability of the logistics engineer to influence the weapon system design; create an acquisition logistics data base management system which can be used by government personnel and contractor personnel; demonstrate the benefits of linking logistics data bases with engineering design data bases, specifically CAD and system test data bases; eliminate the need for paper- intensive logistics documentation

systems; and improve data availability for interfacing with the Air Training Command.

Graphics Standards: There is a general need and a specific one in UDB for graphical output of queries. This means that new applications would have to be written to handle this if the dbms can't. If the dbms can handle graphics there must be a standard interface between dbms and graphics. If the dbms can't handle graphics, the new applications would have to use GKS, etc. to ensure device- independence within UDB and also for interface to other programs.

IRDS, SQL & NDL: They are currently using IDMS and IDD.
NDL & SQLTherefore IRDS and NDL or SQL would be appropriate to enhance compatibility and exchange of data with other systems. There is a need for a graphics interface to IRDS and the appropriate database standard.

Project Title: Product Definition Data Interface

Goals and Objectives: The PDDI program will establish a digital interface between engineering and manufacturing which replaces the engineering part drawing. PDDI objectives are: to evaluate the Initial Graphics Exchange Specification (IGES); to specify manufacturing informations needs from engineering; to create prototype interface to exchange product definition data between dissimilar systems in a factory and to demonstrate the program in a production environment with in-house manufacturing systems and commercial CAD/CAM Systems. PDDI seeks to lower the costs of creating, managing and communicating part descriptions for aircraft systems by allowing the data to be transmitted in a standard, public domain format.

Contact: Lt. Robert A. Carringer

Contact Telephone: (513) 255-6976 AV 785-6976

Contractor: McDonnell/Douglas

Acronym: PDDI

Technical Approach: PDDI addressed the engineering to manufacturing interface for four (4) classes of airframe parts: machined, composite, sheet metal, and turned. The product definition data for these parts will encompass an estimated 90% of the data types describing a typical airframe. The product definition data was identified in a walk through process whereby each manufacturing step required to produce the part was evaluated to identify all product description information used. The information was organized into a data model and file specification. Software for manipulating the information was designed and built.

PDDI successfully demonstrated the transfer of complete product definition data, the portability of the PDDI software, and the use of product definition data in manufacturing. Demonstrations of group technology classification and coding, computer aided process planning, robotic programming, NC programming, composite nesting, and interfacing to commercial CAD/CAM systems were presented to industry in September 1985.

PDDI is being modified to demonstrate the system for exchanging composite and machined part models to Sacramento ALC's composite center and NC shop. Additional demonstrations will occur at Vought's Flexible Machining Cell.

Graphics Standards: Same as for GMAP.

IRDS, SQL & NDL: Same as for GMAP.

Project Title: Integrated Information Support System

Goals and Objectives: To establish and operate a test bed to validate the concept of Integrated Applications supported by an Integrated Information Support System (IISS) in a heterogenous computer and data base environment. In addition, the project is using a set of interim standards and procedures to guide the enhanced design of the IISS. The set of requirements established from 1984 prototype and 1986 production implementation designs will be the basis for enhancements to the IISS. The test bed will serve as a technology transfer vehicle, training facility and a central development site. The test bed realizing the full benefits will also serve as a facility to assist the DoD Community in achieving these benefits faster and with less risk.

Contact: David Judson

Contact Telephone: (513) 255-6976 AV 785-6976

Contractor: Boeing

Acronym: IISS

Technical Approach: It has been estimated that in large U.S. corporations, most of the existing computer applications will be redesigned over the next 10 to 20 years. It is further expected that, because of the rapidly changing computer technology, the construction techniques and operation modes of new application will bear little resemblance to those of existing systems.

The Prime Contractors and coalitions to date have provided a set of five principles as guides in formulating the IISS solution which is extendible for the long-term trends. Individually, each of these principles reflects state-of-the- art technology and has been implemented in the integrated prototype IISS to yield a functionally integrated system. These principles are stated as follows.

1. The IISS is a key mechanism for integration of computerized manufacturing. It defines, controls, and executes actions affecting information among various functionally independent subsystems, based on the use of common data.

2. The IISS employs a coordinated data base approach to support information resource management of various application systems in a closed loop environment within the manufacturing enterprise.

3. The IISS implementation strategy employs several stages of date and application control.

4. The IISS operates as a transaction oriented system responding interactively to user commands.

5. The IISS is a distributed set of heterogenous computer hardware and software systems accessible from geographically dispersed locations.

These principles were inputs to establish a set of requirements and specifications, which resulted in an overall system design developed with capability of both short- and long-term migration. To focus attention on the long-range needs and requirements for IISS, projections beyond 1995 concerning computer systems architectures were researched and established as the baseline. The IISS development has participated in advancing standards in fifteen (15) ANSI and ISO committees and has worked with industry and NBS on communications, a major IISS subsystem, implementing OSI protocols (MAP/TOP).

Over ten (10) major releases of software including methodologies, languages, and compilers have been released since early 1983. The Boeing effort continues the baseline established by four years of General Electric coalition effort ending in 1986. The transition to Boeing will see the system hardened and implemented in production at Boeing and McDonnell Aircraft Co. in 1986 thru 1989. Technology Transfer has been initialized to the computer handware and software vendors.

Graphics Standards: Since IISS provides the integration within a completely homogeneous environment, graphics standards should be used to provide machine, and especially device-independence. GKS, PHIGS, CGM and CGI would all fit in well here.

IRDS, SQL & NDL: IRDS, SQL, and probably NDL are all applicable. (General Dynamics currently using 300 IMS databases on shop floor.) Currently using a "sophisticated" DDS with (IDEF1*) model sitting on top. Neutral DML is SQL based.

Project Title: Engineering Drawing Automated Storage and Retrieval Equipment

Goals and Objectives: The DLA Engineering Drawing Automated Storage and Retrieval Equipment (EDASRE) project is directed toward automating four technical data repositories to support information requests and reprocurement actions. The respositories include the Defense Electronics Supply Center (DESC), Defense General Supply Center (DGSC), Defense Industrial Supply Center (DISC), and Defense Construction Supply Center (DCSC). Repository functions of each of the Supply Centers are established under DoDI 5010.12, Technical Data Management Program. The DLA EDASRE program is comprised of two phases. The first phase provides the Agency with an interim capability to fully automate processing of aperture card files at the four technical data repositories, thus eliminating the need to manually store and retrieve aperture cards in response to customer requests for technical information.

Phase 2 of the EDASRE program will incorporate Agency planning to transition from our Phase 1 interim automated aperture card systems into DSREDS/EDCARS digital storage and retrieval environment for the processing of engineering drawings and reprocurement bidset packages. It is the Agency's intent that upon completion of the EDASRE Phase 2 planning that we will acquire digital capability through any existing DoD contract that may be used as a means for acquisition. Otherwise, we will look toward the DSRED/EDCARS experience in defining digital processing requirements and acquisition criteria before taking any competitive procurement action.

Contact:

Contact Telephone:

Acronym: EDASRE

Technical Approach:

Project Title: Modernized Parts Control Automated Support System

Goals and Objectives: The Defense Logistics Agency Parts Control Automated Support System (PCASS) currently has many manual functions involved in the administration and performance of the DoD Parts Control Program, DoDI 4120.19, MIL-STD 965. This program supports Military Parts Control Advisory Group (MPCAG) operations. The MPCAGs are located at four different Defense Supply Centers (DSCs), which evaluate and make recommendations on parts proposed for use in systems being developed by the DoD and other Federal agencies. The MPCAG's responsibilities are to promote standardization through the use of military specification parts.

Objectives:

1. Provide an on-line database to replace PCASS sequential tape files, allowing near-real-time processing and ad hoc query capability for the military services and industry.

2. Provide for almost 100% growth to 1,000 contracts supported per year, with improved response time.

3. Automate the remaining manual/paper reference files, including status of Mil/Spec standard preparation (over 1,200 per year) and Qualified Products Lists (over 200 lists).

4. Provide telecommunications for system input/output, and for interface with reference files in other DoD and industry ADP systems.

Contact:

Contact Telephone:

Acronym: MPCASS

Technical Approach:

Project Title: Personal Electronic Aid for Maintenance

Goals and Objectives: This is a joint effort with the Army to develop test and evaluate an authoring and electronic portable field maintenance system. The Navy project is focused on the extensive use of graphics displays as troubleshooting aides for use by the maintenance technician. The system is being designed to provide the technician with a checklist of maintenance actions and step-by-step procedures in combined text and graphics format.

Contact:

Contact Telephone:

Acronym: PEAM

Technical Approach:

Project Title: Computer-Based Aide for Troubleshooting

Goals and Objectives: The increasing sophistication of modern Navy weapon systems has resulted in an exponential growth in the technical information required for support. This phenomenal growth often does not include the additional documentation required to support maintenance of the advanced electronic circuitry beyond the point where the automatic test equipment (ATE) and built-in-test (BIT) leave off. Complete reliance on ATE and BIT forces the technician, when ATE and BIT fail, to fault isolate without any additional technical information. While the amount of data is, in itself, a problem, complexity in the presentation of information aggrevates it. Specifically, it is the user's access to an interaction with the technical information that is formidable.

The objective of this project is to define, describe, and evaluate the technical information variables that contribute to effective maintenance performance by: (1) development of a technology base for technical information design and delivery, and (2) design, development, test, and implementation of an intelligent user defined technical information system.

Contact:

Contact Telephone:

Acronym: CBAT

Technical Approach:

Project Title: Navy Integrated CAD-CAM

Goals and Objectives: This project will establish hardware and software system specifications, develop program documentation and execute consolidated acquisition and integrated installation of CAD-CAM systems at principal engineering and industrial activities performing design/maintenance of ships, systems and facilities.

Contact:

Contact Telephone:

Acronym:

Technical Approach:

Project Title: Computer-Aided Integrated Logistic Life Cycle Support (Computer Aided Logistical Support Analysis)

Goals and Objectives: Apply RAM across the spectrum of logistic support analysis and the logistical support analysis record over the life cycle of the acquisition. To provide front end and supportability inputs and considerations to the design and engineering process for the acquisition process.

Using the government owned CALSA (with the MK 50 Lightweight Torpedo Program as a testbed), the first steps have been to record with current primary emphasis on the timely spare provisioning process, reviews and audits.

Follow-on efforts will integrate RAM/CAD on an interactive basis utilizing existent databases for corporate memory to provide front end inputs to the design and engineering process.

Contact:

Contact Telephone:

Acronym: CALSA

Technical Approach:

APPENDIX 6

SHORT AND LONG TERM SOLUTIONS TO RASTER VS VECTOR PROBLEM IN CALS

APPENDIX 6

SHORT AND LONG TERM SOLUTIONS TO RASTER VS VECTOR PROBLEM IN CALS

From the resources investigated (in the contract report [SPAR86]), two major graphics-related opinions emerged, in very clear and consistent statements throughout each project:

1. There is a tremendous backlog of data that must be accessed, manipulated, and outputted, currently in raster format. Much of this data is archived on aperture cards that must be digitally scanned. Data in this format will be a part of the CALS database for a very long time.

2. Many of the current and future graphics data inputs will be originated at a CAD or CAE terminal that has capabilities for manipulating and storing 3D solid product models and/or an image in vector format. The database capabilities and engineering facilities provided by such systems, as well as the current trend towards lower cost, make their use highly desirable for production of the original drawings.

These two facts negate any attempt to determine a single, inclusive format for all CALS graphics data. Any solution must be able to handle both raster and vector types of images.

Vector images can be translated into raster format. In fact, in most cases, during display of the image on a graphics device, they are converted to a raster representation so that the hardware graphics engine can display the image. However, the reverse is not true, given the current state of the art. Although the command-and-parameter format of a vector image, such as "draw a line from point one to point two," can be fairly easily turned into a series of on-off values for a raster display memory, it is much more difficult to recognize that same series of on-off points as a contiguous line. A major problem exists in simply checking the resulting output for validity and completeness. There are several industries that are attempting to resolve this problem, while raster to vector conversion is cost effective for drawing modification in some cases, currently there is no cost effective solution for document archival.

1. Short Term Solution

The least common denominator in terms of technical difficulty is the raster format of documents. The existence of raster-scanned documents does not preclude the capture of hand-drawn images, CAD-generated documents or alphanumeric databases. Therefore, until standards are available to handle vector-formatted data, it is recommended that a raster based CALS system be standardized.

In such a system, CALS would store a raster facsimile of all DOD documents. Any document could then be located and transmitted anywhere in the world in seconds for display or printing. As standards for more complex data elements become available, CALS can be modified to store, retrieve, and integrate them. CALS could also store and retrieve binary files that do not conform to a standard, although it must be recognized that such files will be of limited value after a time, due to inevitable modification of the systems that created them.

Raster image databases should be easier to manage than text, graphic, or inventory databases, since raster images contain fewer internal structures. This means that raster databases will not require distributed database management systems, enhancing the possibility of fast implementation.

Although much discussion of the surveyed projects has revolved around the problems of perceived slowness of display, and cost and amount of storage required by using only raster as the database format, current technological trends are towards cheaper storage and faster decompression and raster display devices.

This short-term solution allows for the accomodation of current archival images as well as providing for the gradual development of the long-term solution described in the following section, without impacting the day by day performance of the projects.

2. Long Term Solution

CALS should support both raster and vector formatted data. The proposed long term solution is, while continuing to keep raster scanned documents in raster format, to migrate all incoming data to vector format. This means to encourage originals to be prepared on CAD systems, with a standard format for transferring and storing the vector version of each image. This format is already defined: it is the CGM standard format for the graphics operations, with a backup of IGES/PDES for the product definition data requirements.

In addition to using the established CGM and IGES/PDES standards for data storage, there is a need for standard interfaces to the target display devices. Many situations, from the soldier in the field attempting to upgrade or maintain some equipment and the logistician in the office updating some information in a document, to management personnel briefing DOD Chiefs, require diverse devices for display and input to the data management programs that provide these capabilities. The emerging CGI standard, in addition to the GKS and PHIGS application-level standards, can be used optimally to provide the required device-independence. Another benefit of using these standards is programmer portability for application software development or modification, and the cost benefits of increased availability of Off-The-Shelf hardware and software.

There will be, then, a standard storage base of images, in vector format if possible, that may be indexed, retrieved, and modified if necessary. The primary representation of the image, however, will remain the raster formatted version. The last step of every retrieval will be a rasterization step.

This solution provides both a quick, global access to documents and images, from the raster version of the image, and flexible storage of the same image in a format that is accepted from contractors in a standard way. The perceived configuration management problems with a dual representation method have actually been dealt with over the years as multiple versions of documents were created and maintained. Older versions of a document can be controlled in raster format while newer versions, rewritten or amended in alphanumeric or vector format, can be controlled in their new format. Controlling these multiple versions of documents is a capability that CALS must have whether the versions are in the same format or different formats. As described above, reformatting such as raster to vector conversion, requires human input while the reverse vector to raster conversion can be done automatically. Reformatting requiring human input should be controlled as a new revision while automatic reformatting should be considered part of the output process. Because the output process can have several conversions with intermediate buffers, CALS must be designed to accomodate multiple automatically reformatted versions of a document. If automatic reformatting requires extensive computing, the intermediate results may be retained until the source document is changed.

CALS must also accomodate multiple identical copies of each document to provide spatial diversity for survivability of the combat readiness document-base. CALS must have an integrated, on-line backup capability.

3. Conclusion

CALS must support raster images because almost all documentation either lacks an alphanumeric/vector form, or, if one is used, it is nonstandard. CALS must support standard alphanumeric and vector storage formats to facilitate full text searches and CAD input/modification. Choosing to remain with one or the other format exclusively either would result in losing the capability to access archived raster images or would result in losing the design and solid model information and the ability to search for text strings.

The raster-only short term solution is suggested, since it can be applied early and easily and cheaply. In addition, a successful raster CALS system would almost certainly be able to inspire a budget for additional vector capabilities.

For the long term, however, CALS must also provide portability of software tools and ease of transfer of images and product data from one system to another. For this reason, CALS must accept documents in vector and alphanumeric formats as soon as standards for these formats are established.

APPENDIX 7

TOP/MAP COMPUTER GRAPHICS METAFILE (CGM) APPLICATION PROFILE (AP)

Title: TOP/MAP Computer Graphics Metafile (CGM) Application Profile (AP)

Contributor: Frank R. Dawson (Editor)
McDonnell Douglas Corporation

Purpose: Text of an Application Profile defining the TOP/MAP Graphics Technical Subcommittee's Recommendation for support of the CGM as the interchange format for computer graphics picture description information in an OSI environment.

Date: 7 August 1986

0 Introduction

The purpose of this document is to define the TOP/MAP Application Profile for support of the ISO 8632, Information Processing Systems Computer Graphics Metafile for the Storage and Transfer of Picture Description Information, the CGM. An application profile defines the conformance characteristics or permissible combinations for all possible data streams that conform to that profile. In addition, an application profile may define additional requirements for transmitting, receiving, interpreting and handling valid CGM data streams. The definition of such implementation constraints is usually outside the scope of an ISO standard. However, such application profiles are required and necessary to insure uniform implementation of such standards, especially where interchange in an open system environment is concerned.

The application profile specifies the conformance to the CGM in terms of **PERMISSIBLE**, **BASIC**, **NONBASIC**, and **DEFAULT** values. Permissible values are the range of values for CGM elements as specified in ISO 8632. Basic values are the range of permissible values that are mandatory for conformance to this application profile. Nonbasic values are the remainder of permissible values for CGM elements. The default values for CGM elements are the implicit initial values that are assumed for each parameter. Default values are explicitly overridden by the Metafile Description Elements, Picture Description Elements, Control Elements, and Attribute Elements.

This TOP/MAP Application Profile is the basis of the TOP Specification for interchange of computer graphics picture description information. This application profile can, in the future, be supplemented by additional TOP/MAP CGM Application Profiles.

1 Scope

This TOP/MAP CGM Application Profile defines the CGM implementation that is recommended by TOP/MAP for interchanging computer graphics picture information. CGM implementations that conform to this application profile will be able to be integrated into other TOP/MAP Application Processes (i.e., TOP/MAP Compound Document Interchange Format).

2 References

These references are applicable to this document.

NBS Special Publication 424 - April 1976, Hershey Fonts.
ISO 646, Information Processing - 7-bit Coded Character Set for Information Interchange.
ISO 2022, Information Processing - 7-bit and 8-bit Coded Characters Set Code Extension Techniques.

ISO 7942, Information Processing Systems - Computer Graphics Functional Specification of the Graphical Kernel System (GKS).

ISO 8613, Information Processing - Text and Office Systems - Office Document Architecture (ODA) and Interchange Format.

ISO 8632, Information Processing Systems - Computer Graphics Metafile (CGM) for the Storage and Transfer of Picture Description Information.

3 Definitions

APPLICATION PROFILE - A specification that defines the use of an International Standard, with a definition of all possible data streams that conform to that profile. An application profile insures interoperability of implementations of an International Standard.

BASIC VALUE - The subset of permissible values for parameters of a CGM element that are mandatory for conformance to this application profile.

CGM MI - A CGM metafile input workstation.

CGM MO - A CGM metafile output workstation.

COMPOUND DOCUMENT - A digital analog of a document containing more than one component objects, such as character, computer graphics, image or facsimile data.

COMPOUND DOCUMENT INTERCHANGE FORMAT - The specification for a mechanism for storing and transferring a compound document. Refer to ISO 8613.

COMPUTER GRAPHICS INTERFACE (CGI) - The specification for interface techniques with graphical devices.

COMPUTER GRAPHICS METAFILE (CGM) - The specification for a mechanism for storing and transferring picture description information. Refer to ISO 8632.

DATA INTERFACE - An interface between software modules or devices comprising one or more operation codes and data; as contrasted with a subroutine call interface.

DEFAULT VALUE - The implicit value for a parameter of a CGM element (e.g., default Metafile Name in Begin Metafile is the null string).

DEVICE DRIVER - The device dependent part of a graphics system which supports a physical device. The device driver generates device class specific output.

GRAPHICAL KERNEL SYSTEM - A standardized application programmer's interface to graphics systems. Refer to ISO 7942.

METAFILE - Used synonymous with CGM. A mechanism for retaining and transporting graphical data and control information. This information contains a device independent description of one or more pictures.

METAFILE GENERATOR - Used synonymous with CGM Generator. The software or hardware that creates a picture or conveys information in the CGM representation.

METAFILE INTERPRETER - Used synonymous with CGM Interpreter. The software or hardware that reads the CGM and interprets the contents.

NONBASIC VALUE - The set of permissible values for a parameter of a CGM element other than the basic values.

PERMISSIBLE VALUE - The range of valid values for a parameter of a CGM element as specified in ISO 8632.

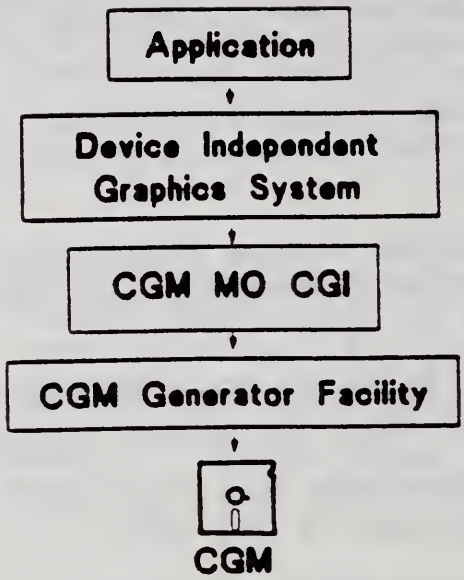
VIRTUAL DEVICE - An idealized computer graphics device that presents a set of graphics capabilities to graphics software or systems via the CGI.

NOTE: Refer to clause 3 of ISO 8632 and ISO 7942 for further definitions of computer graphics terms.

4 Architectural Concepts

The CGM is designed to be usable and useful to a wide range of applications, graphics systems, and devices or workstations. The CGM is graphics system independent, as well as device independent. The CGM is created by a CGM Generator. The CGM Generator resides at the level of the device driver and is invoked by the application callable layer. The CGM Generator can be used to record device independent picture descriptions, conceptually in parallel with the presentation of images on actual devices. Figure 4.1 illustrates the reference model for creation of the CGM.

Figure 4.1: CGM Generator Reference Model



The CGM is designed to be interpreted either by a special application program, that in turn invokes a device independent graphics system to render the CGM. As well, it may be interpreted by an application callable, device independent graphics system through the use of CGM metafile input CGI. Figure 4.2 illustrates the reference model for interpretation of the CGM.

Figure 4.2: CGM Interpreter Reference Model

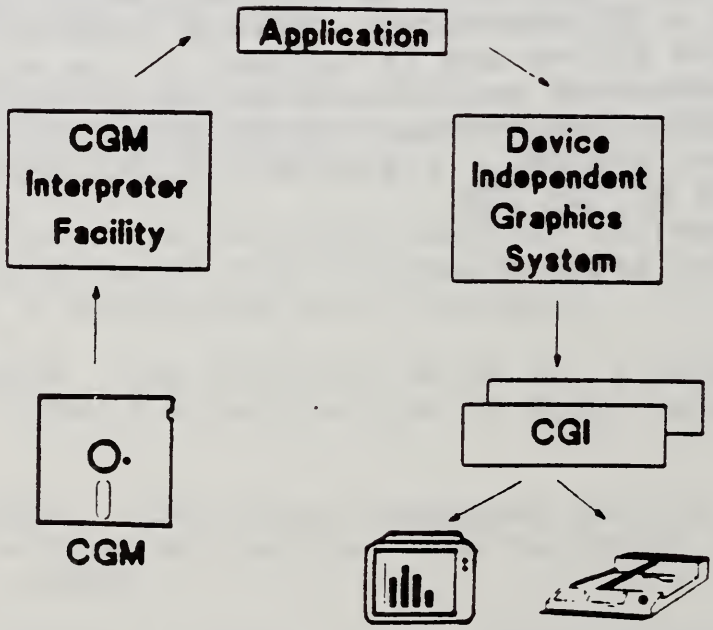


Figure 4.3: Alternate CGM Interpreter Reference Model

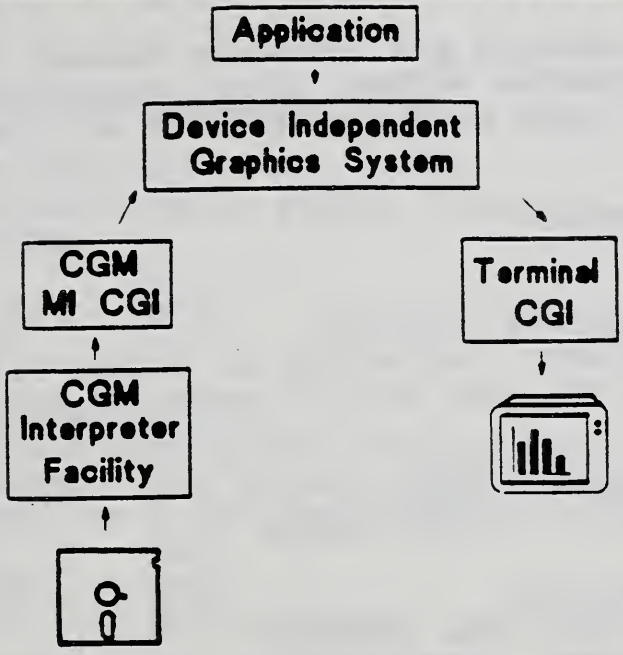


Figure 4.3 illustrates the alternative reference model for interpretation of the CGM.

The GKS may be the application callable, device independent graphics system that is used in these reference models. The GKS standard, however, does not specifically refer to the CGM any more that it does to any other specific class of graphics device. Sections 9 and 10 define the mapping between GKS and CGM as supported by this application profile. The definition of the conformance for this mapping will permit a specific GKS based application to create the same CGM representation on any vendor's implementation of the GKS.

5 Encoding

The CGM standard defines the form (syntax) and the functional behavior (semantics) of the ordered set of metafile elements. There are three different encodings of the CGM that have been standardized. These include the Clear Text Encoding, Character Encoding, and Binary Encoding. This application profile specifies that the CGM will be encoded according to the Binary Encoding defined in ISO 8632/3.

The basic form for the command header and string parameter header is the long form. The nonbasic form for the command header and string parameter header is either the short or long form. The long form of the command header is detailed in ISO 8632/3, subclause 4.4, pages 7-9. The long form of the string parameter header is detailed in ISO 8632/3, clause 6, page 17, note 6.

6 Metafile Constraints

Each of the elements are listed by element class. For each element, the permissible values, basic values, nonbasic values, and default values for each of the parameters are listed.

6.1 Delimiter Elements

Element: NOOP
Parameter 1: Pad string (String)
- Permissible Values: Any string of octets with a length greater than or equal to 0 and less than or equal to the Maximum String Length.
- Basic Values: Any
- Nonbasic Values: None
- Default Values: Null string

Element: Begin Metafile
Parameter 1: Metafile name (String).
- Permissible Values: Any string of characters with a length greater than or equal to 0 and less than or equal to the Maximum String Length.
- Basic Values: Any
- Nonbasic Values: None
- Default Values: Null string

Element: End Metafile
Parameter 1: None
- Permissible Values: N/A
- Basic Values: N/A
- Nonbasic Values: N/A
- Default Values: N/A

Element: Begin Picture
Parameter 1: Picture name (String).
- Permissible Values: Any string of characters with a length greater than or equal to 0 and less than or equal to the Maximum String Length.
- Basic Values: Any
- Nonbasic Values: None
- Default Values: Null string

Element: Begin Picture Body
Parameter 1: None
- Permissible Values: N/A
- Basic Values: N/A
- Nonbasic Values: N/A
- Default Values: N/A

Element: End Picture
Parameter 1: None
- Permissible Values: N/A

- Basic Values: N/A
- Nonbasic Values: N/A
- Default Values: N/A

6.2 Metafile Descriptor Elements

Element: Metafile Version
Parameter 1: Metafile version number (Integer).
- Permissible Values: 1
- Basic Values: 1
- Nonbasic Values: None
- Default Values: 1

Element: Metafile Description
Parameter 1: Metafile descriptive string (String).
- Permissible Values: Any string of characters with a length greater than or equal to 0 and less than the Maximum String Length.
- Basic Values: Any
- Nonbasic Values: None
- Default Values: Null string

Element: VDC Type
Parameter 1: VDC type (Integer)
- Permissible Values: 0, integer VDC
  1, real VDC
- Basic Values: 0, 1
- Nonbasic Values: None
- Default Values: 0

Element: Integer Precision
Parameter 1: Integer precision (Integer)
- Permissible Values: 8, 8-bit
  16, 16-bit
  24, 24-bit
  32, 32-bit
- Basic Values: 16
- Nonbasic Values: Any
- Default Values: 16

Element: Real Precision
Parameter 1: Representation form (Integer)
- Permissible Values: 0, IEEE 754 floating point format
  1, Fixed point format
- Basic Values: 1
- Nonbasic Values: Any
- Default Values: 1

Parameter 2: Exponent field width (Integer)
- Permissible Values: 9, 8-bit with sign
  12, 11-bit with sign
  16, 15-bit with sign
  32, 31-bit with sign
- Basic Values: 16
- Nonbasic Values: Any

- **Default Values:** 16

Parameter 3: Fraction field width (Integer)
- **Permissible Values:** 23, 23-bit
  52, 52 bit
  16, 16-bit
  32, 32-bit
- **Basic Values:** 16
- **Nonbasic Values:** Any
- **Default Values:** 16

Element: Index Precision
Parameter 1: Index precision (Integer)
- **Permissible Values:** 8, 8-bit
  16, 16-bit
  24, 24-bit
  32, 32-bit
- **Basic Values:** 16
- **Nonbasic Values:** Any
- **Default Values:** 16

Element: Color Precision
Parameter 1: Color precision (Integer)
- **Permissible Values:** 8, 8-bit
  16, 16-bit
  24, 24-bit
  32, 32-bit
- **Basic Values:** 8, 16
- **Nonbasic Values:** Any
- **Default Values:** 16 **Note:** This element must be explicitly specified in the Metafile Description with this default value.

Element: Color Index Precision
Parameter 1: Color index precision (Integer)
- **Permissible Values:** 8, 8-bit
  16, 16-bit
  24, 24-bit
  32, 32-bit
- **Basic Values:** 16
- **Nonbasic Values:** Any
- **Default Values:** 16 **Note:** This element must be explicitly specified in the Metafile Description with this default value.

Element: Maximum Color Index
Parameter 1: Maximum color index (Index)
- **Permissible Values:** Any positive index value in the range specified by Color Precision.
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 255

Element: Color Value Extent
Parameter 1: Minimum color value (RGB-tuple)
- **Permissible Values:** Any RGB-tuple of positive integers in range specified by Color Precision.

- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** (0,0,0)

Parameter 2: Maximum color value (RGB-tuple)
- **Permissible Values:** Any RGB-tuple of positive integers in range specified by Color Precision.
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** (255,255,255)

Element: Metafile Elements List

Parameter 1: Number of elements specified (Integer)
- **Permissible Values:** Integers in range specified by Integer Precision.
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 1

Parameter 2: List of metafile element class/identifier (Index-pair array)
- **Permissible Values:** Ordered pair of indices in range of element class/identifier pairs as specified ISO 8613/3, Annex C.
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** (-1,1), Drawing-plus-control set

Element: Metafile Defaults Replacement

Parameter 1: List of metafile elements (Metafile Elements)
- **Permissible Values:** Any set of Picture Descriptor, Control, or Attribute Elements with basic values
- **Basic Values:** Any
- **Nonbasic Values:** Any set of Picture Descriptor, Control, or Attribute Elements with nonbasic values
- **Default Values:** The default is a Metafile Defaults Replacement element composed of (1) Text Precision Element specifying a text precision of 2 (stroke) and (2) Color Table specifying 256 indices. The indices are defined as follows:

```
Index 0 = (0,0,0)          - Nominal Background
      1 = (255,255,255)    - Nominal Foreground
      2 = (255,0,0)        - Red
      3 = (0,255,0)        - Green
      4 = (0,0,255)        - Blue
      5 = (255,255,0)      - Yellow
      6 = (255,0,255)      - Magenta
      7 = (0,255,255)      - Cyan
      8 - 255 are a replication of indices 0 - 7
```

Element: Font List

Parameter 1: List of font names (string arrays)
- **Permissible Values:** Any number of strings in the range [0,Maximum String Array Length], each with an arbitrary string of n octets, where 0 <= n <= Maximum String Length.
- **Basic Values:** Any
- **Nonbasic Values:** None

- Default Values: Name of one font capable of representing the standard national character (i.e., ANS X3.4) set based on ISO 646

Element: Character Set List
Parameter 1: Character set type
- Permissible Values: 0, 94-character G-sets
  1, 96-character G-sets
  2, 94-character multibyte G-set
  3, 96-character multibyte G-set
  4, Complete
- Basic Values: 0
- Nonbasic Values: Any
- Default Values: 0

Parameter 2: Designation sequence tail
- Permissible Values: An arbitrary string of n octets. 0 <= n <= Maximum String Length.
- Basic Values: The three characters Esc 2/8 4/2 designating ANS X3.4 (7-bit ASCII).
- Nonbasic Values: Any
- Default Values: The three characters Esc 2/8 4/2 designation ANS X3.4 (7-bit ASCII). Note: This element must be explicitly specified in the Metafile Description with this default value.

Element: Character Coding Announcer
Parameter 1: Character coding announcer (Integer)
- Permissible Values: 0, Basic 7-bit
  1, Basic 8-bit
  2, Extended 7-bit
  3, Extended 8-bit
  Negative values for private use
- Basic Values: 0,1
- Nonbasic Values: None
- Default Values: 0

6.3 Picture Descriptor Elements

Element: Scaling Mode
Parameter 1: VDC scaling mode (Integer)
- Permissible Values: 0, Abstract scaling
  1, Metric scaling
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 0

Parameter 2: Metric scaling factor in millimeters (Real)
- Permissible Values: Any real value in range specified by the real type and real precision. Ignored if scaling mode is abstract.
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 25.4

Element: Color Selection Mode
Parameter 1: Color selection mode (Integer)
- Permissible Values: 0, Indexed color mode

1, Direct color mode
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 0

Element: Line Width Specification Mode
Parameter 1: Line width specification mode (Integer)
- Permissible Values: 0, Absolute
1, Scaled
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 1

Element: Marker Size Specification Mode
Parameter 1: Marker size specification mode (Integer)
- Permissible Values: 0, Absolute
1, Scaled
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 1

Element: Edge Width Specification Mode
Parameter 1: Edge width specification mode (Integer)
- Permissible Values: 0, Absolute
1, Scaled
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 1

Element: VDC Extent
Parameter 1: First point (Point)
- Permissible Values: Any value for x and y VDC in range specified by VDC Type and VDC Precision
- Basic Values: Any
- Nonbasic Values: None
- Default Values: (0,0) if VDC Type is Integer
(0.0,0.0) if VDC Type is Real
Parameter 2: Second point (Point)
- Permissible Values: Any value for x and y VDC in range specified by VDC Type and VDC Precision
- Basic Values: Any
- Nonbasic Values: None
- Default Values: (32767,32767) if VDC Type is Integer
(0.9999..,0.9999..) if VDC Type is Real

Element: Background Color
Parameter 1: Direct background color (RGB-tuple)
- Permissible Values: RGB-tuple of positive integers in range specified by color precision.
- Basic Values: Any
- Nonbasic Values: None
- Default Values: (0,0,0) or Minimum Color Value

6.4 Control Elements

Element: VDC Integer Precision
Parameter 1: VDC integer precision (Integer)
- **Permissible Values:** 16, 16-bit
  24, 24-bit
  32, 32-bit
- **Basic Values:** 16
- **Nonbasic Values:** Any
- **Default Values:** 16

Element: VDC Real Precision
Parameter 1: Representation form (Integer)
- **Permissible Values:** 0, IEEE 754 floating point format
  1, Fixed point format
- **Basic Values:** 1
- **Nonbasic Values:** Any
- **Default Values:** 1

Parameter 2: Exponent field width (Integer)
- **Permissible Values:** 9, 8-bit with sign
  12, 11-bit with sign
  16, 15-bit with sign
  32, 31-bit with sign
- **Basic Values:** 16
- **Nonbasic Values:** Any
- **Default Values:** 16

Parameter 3: Fraction field width (Integer)
- **Permissible Values:** 23, 23-bit
  52, 52 bit
  16, 16-bit
  32, 32-bit
- **Basic Values:** 16
- **Nonbasic Values:** Any
- **Default Values:** 16

Element: Auxiliary Color
Parameter 1: Auxiliary Color (RGB-tuple or Color Index)
- **Permissible Values:** Any valid RGB-tuple or color index in the range specified by Color Precision (RGB-tuple) or Color Index Precision (color index).
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** Default background color, when Color Specification Mode is direct. Color Table index 0, when Color Specification Mode is indexed.

Element: Transparency
Parameter 1: Transparency
- **Permissible Values:** 0, Off; auxiliary color background is required
  1, On; transparent background is required
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 1

Element: Clip Rectangle
Parameter 1: First point (Point)
- Permissible Values: Any value for x and y VDC in range specified by VDC Type and VDC Precision
- Basic Values: Any
- Nonbasic Values: None
- Default Values: Same as first point of VDC Extent

Parameter 2: Second point (Point)
- Permissible Values: Any value for x and y VDC in range specified by VDC Type and VDC Precision
- Basic Values: Any
- Nonbasic Values: None
- Default Values: Same as second point of VDC Extent

Element: Clip Indicator
Parameter 1: Clip indicator (Integer)
- Permissible Values: 0, Off
  1, On
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 1

6.5 Graphical Primitives

For all elements in this class, parameters can have any of the permissible values as specified in ISO 8632. The basic values are any of the permissible values that are restricted by the Environmental Constraints (i.e., Maximum Point Array Length, Maximum String Length, etc.). There are no nonbasic values. The default values are not applicable.

6.6 Attribute Elements

Element: Line Bundle Index
Parameter 1: Line bundle index (Index)
- Permissible Values: Any positive index in the range specified by the Index Precision
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 1

Element: Line Type
Parameter 1: Line type (Integer)
- Permissible Values: 1, Solid
  2, Dash
  3, Dot
  4, Dash-dot
  5, Dash-dot-dot
  Any negative integer in the range specified by the Integer Precision for private use
- Basic Values: Any
- Nonbasic Values: None

- **Default Values:** 1

Element: Line Width
Parameter 1: Line width (VDC or Real)
- **Permissible Values:** Any valid VDC in the range specified by VDC Type and VDC Precision if Line Width Specification Mode is absolute or any valid Real in the range specified by Real Precision if Line Width Specification Mode is scaled.
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 0.001*Length of the longest side of rectangle defined by VDC extent for absolute; 1.0 for scaled

Element: Line Color
Parameter 1: Line color (RGB-tuple or Color Index)
- **Permissible Values:** Any valid RGB-tuple or color index in the range specified by Color Precision (RGB-tuple) or Color Index Precision (color index). RGB-tuple values not to exceed Maximum Color Value. Color representation must match Color Representation Mode.
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** Maximum Color Value, when Color Specification Mode is direct. Color Table index 1, when Color Specification Mode is indexed.

Element: Marker Bundle Index
Parameter 1: Marker bundle index (Index)
- **Permissible Values:** Any positive index in the range specified by the Index Precision
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 1

Element: Marker Type
Parameter 1: Marker type (Integer)
- **Permissible Values:** 1, Dot
  2, Plus
  3, Asterisk
  4, Circle
  5, Cross
  Any negative integer in the range specified by Integer Precision for private use
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 3

Element: Marker Size
Parameter 1: Marker Size (VDC or Real)
- **Permissible Values:** Any valid VDC in the range specified by VDC Type and VDC Precision if Marker Size Specification Mode is absolute or any valid Real in the range specified by Real Precision if Marker Size Specification Mode is scaled.
- **Basic Values:** Any

- **Nonbasic Values:** None
- **Default Values:** 0.01*Length of the longest side of rectangle defined by VDC extent for absolute; 1.0 for scaled

Element: Marker Color
Parameter 1: Marker color (RGB-tuple or Color Index)
- **Permissible Values:** Any valid RGB-tuple or color index in the range specified by Color Precision (RGB-tuple) or Color Index Precision (color index). RGB-tuple values not to exceed Maximum Color Value. Color representation must match Color Representation Mode.
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** Maximum Color Value, when Color Specification Mode is direct. Color Table index 1, when Color Specification Mode is indexed.

Element: Text Bundle Index
Parameter 1: Text bundle index (Index)
- **Permissible Values:** Any positive index value in the range specified by the Index Precision
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 1

Element: Text Font Index
Parameter 1: Text font index (Index)
- **Permissible Values:** Any positive index value in the range specified by the Index Precision
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 1

Element: Text Precision
Parameter 1: Text precision (Integer)
- **Permissible Values:** 0, String
  1, Character
  2, Stroke
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 2

Element: Character Expansion Factor
Parameter 1: Character expansion factor (Real)
- **Permissible Values:** Any positive real value in the range specified by the Real Precision
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 1.0

Element: Character Spacing
Parameter 1: Character spacing (Real)
- **Permissible Values:** Any real value in the range specified by the Real Precision

- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 0.0

Element: Text Color

Parameter 1: Text color (RGB-tuple or Color Index)

- **Permissible Values:** Any valid RGB-tuple or color index in the range specified by Color Precision (RGB-tuple) or Color Index Precision (color index). RGB-tuple values not to exceed Maximum Color Value. Color representation must match Color Representation Mode.
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** Maximum Color Value, when Color Specification Mode is direct. Color Table index 1, when Color Specification Mode is indexed.

Element: Character Height

Parameter 1: Character height (VDC)

- **Permissible Values:** Any VDC value in the range specified by the VDC Type and VDC Precision
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 0.01*Length of the longest side of rectangle defined by VDC extent

Element: Character Orientation

Parameter 1: Character up vector x component (VDC)

- **Permissible Values:** Any VDC value in the range specified by the VDC Type and VDC Precision
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 0 for VDC Type Integer; 0.0 for VDC Type Real

Parameter 2: Character up vector y component (VDC)

- **Permissible Values:** Any VDC value in the range specified by the VDC Type and VDC Precision
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 1 for VDC Type Integer; 1.0 for VDC Type Real

Parameter 3: Character base vector x component (VDC)

- **Permissible Values:** Any VDC value in the range specified by the VDC Type and VDC Precision
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 1 for VDC Type Integer; 1.0 for VDC Type Real

Parameter 4: Character base vector y component (VDC)

- **Permissible Values:** Any VDC value in the range specified by the VDC Type and VDC Precision
- **Basic Values:** Any
- **Nonbasic Values:** None

- Default Values: 0 for VDC Type Integer; 0.0 for VDC Type Real

Element: Text Path
Parameter 1: Text path (Integer)
- Permissible Values: 0, Right
  1, Left
  2, Up
  3, Down
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 0

Element: Text Alignment
Parameter 1: Horizontal alignment (Integer)
- Permissible Values: 0, Normal
  1, Left
  2, Center
  3, Right
  4, Continuous horizontal
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 0

Parameter 2: Vertical alignment (Integer)
- Permissible Values: 0, Normal
  1, Top
  2, Cap
  3, Half
  4, Base
  5, Bottom
  6, Continuous vertical
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 0

Parameter 3: Continuous horizontal alignment (Real)
- Permissible Values: Any real value in the range specified by the Real Precision
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 0.0

Parameter 4: Continuous vertical alignment (Real)
- Permissible Values: Any real value in the range specified by the Real Precision
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 0.0

Element: Character Set Index
Parameter 1: Character set index (Index)
- Permissible Values: Any positive index value in the range specified by the Index Precision
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 1

Element: Alternate Character Set Index
Parameter 1: Alternate character set index (Index)
- Permissible Values: Any positive index value in the range specified by the Index Precision
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 1

Element: Fill Bundle Index
Parameter 1: Fill bundle index (Index)
- Permissible Values: Any positive index value in the range specified by the Index Precision
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 1

Element: Interior Style
Parameter 1: Interior style (Integer)
- Permissible Values: 0, Hollow
  1, Solid
  2, Pattern
  3, Hatch
  4, Empty
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 0

Element: Fill Color
Parameter 1: Fill Color (RGB-tuple or Color Index)
- Permissible Values: Any valid RGB-tuple or color index in the range specified by Color Precision (RGB-tuple) or Color Index Precision (color index). RGB-tuple values not to exceed Maximum Color Value. Color representation must match Color Representation Mode.
- Basic Values: Any
- Nonbasic Values: None
- Default Values: Maximum Color Value, when Color Specification Mode is direct. Color Table index 1, when Color Specification Mode is indexed.

Element: Hatch Index
Parameter 1: Hatch index (Index)
- Permissible Values: 1, Horizontal
  2, Vertical
  3, Positive slope
  4, Negative slope
  5, Combined vertical and horizontal slant
  6, Combined left and right slant
  Negative for private use
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 1

Element: Pattern Index

Parameter 1: Pattern index (Index)
- Permissible Values: Any positive index value in the range specified by the Index Precision
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 1

Element: Edge Bundle Index
Parameter 1: Edge bundle index (Index)
- Permissible Values: Any positive index in the range specified by the Index Precision
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 1

Element: Edge Type
Parameter 1: Edge type (Integer)
- Permissible Values: 1, Solid
2, Dash
3, Dot
4, Dash-dot
5, Dash-dot-dot
Any negative integer in the range specified by the Integer Precision for private use
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 1

Element: Edge Width
Parameter 1: Edge width (VDC or Real)
- Permissible Values: Any valid VDC in the range specified by VDC Type and VDC Precision if Edge Width Specification Mode is absolute or any valid Real in the range specified by Real Precision if Edge Width Specification Mode is scaled.
- Basic Values: Any
- Nonbasic Values: None
- Default Values: 0.001*Length of the longest side of rectangle defined by VDC extent for absolute; 1.0 for scaled

Element: Edge Color
Parameter 1: Edge color (RGB-tuple or Color Index)
- Permissible Values: Any valid RGB-tuple or color index in the range specified by Color Precision (RGB-tuple) or Color Index Precision (color index). RGB-tuple values not to exceed Maximum Color Value. Color representation must match Color Representation Mode.
- Basic Values: Any
- Nonbasic Values: None
- Default Values: Maximum Color Value, when Color Specification Mode is direct. Color Table index 1, when Color Specification Mode is indexed.

Element: Edge Visibility
Parameter 1: Edge visibility

- **Permissible Values:** 0, Off
  1, On
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 0

Element: Fill Reference Point

Parameter 1: Fill reference point (Point)

- **Permissible Values:** Any point whose x and y components are VDC in the range specified by VDC Type and VDC Precision
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** First point defining the VDC Extent

Element: Pattern Table

Parameter 1: Pattern table index (Index)

- **Permissible Values:** Any positive index value in the range specified by Index Precision
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 1

Parameter 2: Dimension of color array in direction of the Pattern Size width vector (Integer)

- **Permissible Values:** Any positive integer in the range specified by Integer Precision
- **Basic Values:** 1 <= dx <= Maximum Color Array Dimension
- **Nonbasic Values:** None
- **Default Values:** 1

Parameter 3: Dimension of color array in direction of the Pattern Size height vector (Integer)

- **Permissible Values:** Any positive integer in the range specified by Integer Precision
- **Basic Values:** 1 <= dy <= Maximum Color Array Dimension
- **Nonbasic Values:** None
- **Default Values:** 1

Parameter 4: Local color precision (Integer)

- **Permissible Values:** 0, Default color precision
  1, 1-bit
  2, 2-bit
  4, 4-bit
  8, 8-bit
  16, 16-bit
  24, 24-bit
  32, 32-bit
- **Basic Values:** 0, 1, 8, or 16 if Color Representation Mode is indexed. 0, 8, or 16 if Color Representation Mode is direct.
- **Nonbasic Values:** Any
- **Default Values:** 0

Parameter 5: Pattern definition (RGB-tuple array or Color Index array)

- **Permissible Values:** Any valid array of RGB-tuple or color index in the range specified by Color Precision (RGB-tuple) or Color Index Precision (color index). RGB-tuple values not to exceed Maximum

Color Value. Color representation must match Color Representation Mode.
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** Maximum Color Value, when Color Specification Mode is direct. Color Table index 1, when Color Specification Mode is indexed.

Element: Pattern Size
Parameter 1: Pattern height vector x component (VDC)
- **Permissible Values:** Any VDC value in the range specified by the VDC Type and VDC Precision
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 0 if VDC Type is Integer; 0.0 if VDC Type is Real

Parameter 2: Pattern height vector y component (VDC)
- **Permissible Values:** Any VDC value in the range specified by the VDC Type and VDC Precision
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** Height of the VDC Extent

Parameter 3: Pattern width vector x component (VDC)
- **Permissible Values:** Any VDC value in the range specified by the VDC Type and VDC Precision
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** 0 if VDC Type is Integer; 0.0 if VDC Type is Real

Parameter 4: Pattern width vector y component (VDC)
- **Permissible Values:** Any VDC value in the range specified by the VDC Type and VDC Precision
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** Width of the VDC Extent

Element: Color Table
Parameter 1: Starting color table index (Color Index)
- **Permissible Values:** Any positive value in the range specified by the Color Index Precision
- **Basic Values:** 1 <= color index <= Maximum Color Index
- **Nonbasic Values:** None
- **Default Values:** 1

Parameter 2: List of direct color values (RGB-tuple array)
- **Permissible Values:** Any array less than or equal to the Maximum Color Array Dimension containing any three-tuple of positive integers in the range specified by the Color Precision and less than the Maximum Color Value.
- **Basic Values:** Any
- **Nonbasic Values:** None
- **Default Values:** One element (255,255,255)

Element: Aspect Source Flags

Parameter 1: List of pairs of ASF type and ASF values (Integer-pair array)
- Permissible Values: As in ISO 8632
- Basic Values: Any
- Nonbasic Values: None
- Default Values: All ASF types set to individual

6.7 Escape Elements

Element: Escape
Parameter 1: Identifier (Integer)
- Permissible Values: Any integer in the range specified by Integer Precision
- Basic Values: Any
- Nonbasic Values: None
- Default Values: N/A

Parameter 2: Escape data record (String array)
- Permissible Values: Any array less than or equal to the Maximum String Array Length containing any string less than or equal to the Maximum String Length.
- Basic Values: Any
- Nonbasic Values: None
- Default Values: N/A

6.8 External Elements

Element: Message
Parameter 1: Action-required flag (Integer)
- Permissible Values: 0, No action
  1, Action
- Basic Values: Any
- Nonbasic Values: None
- Default Values: N/A

Parameter 2: Message string (String)
- Permissible Values: Any string less than or equal to the Maximum String Length.
- Basic Values: Any
- Nonbasic Values: None
- Default Values: N/A

Element: Application Data
Parameter 1: Identifier (Integer)
- Permissible Values: Any integer in the range specified by Integer Precision
- Basic Values: Any
- Nonbasic Values: None
- Default Values: N/A

Parameter 2: Application data record (String array)
- Permissible Values: Any array less than or equal to the Maximum String Array Length containing any string less than or equal to the Maximum String Length.
- Basic Values: Any

- Nonbasic Values: None
- Default Values: N/A

7 Environmental Constraints

This section describes Environmental Constraints or implementation dependencies that are mandatory for conformance to this application profile. Normalizing such implementation practices, defaults, and options will facilitate uniform generation and interpretation of the CGM.

7.1 General Guidelines For Elements

This section is meant to augment ISO 8632/1, subclause D.4.

Name: Color Table
Comments: The Color Table attribute element has an indeterminate effect when it appears in a picture, subsequent to any graphical primitive elements. The Color Table attribute element should appear prior to any graphical primitive elements to insure that interpreting systems without dynamic color update capabilities can render the intended effect.

7.2 Implementation Guidelines

This section is meant to augment ISO 8632/1, subclause D.5 and ISO 8632/3, clause 8.

Name: Maximum Color Array Dimension
Description: The basic value for the number of color values that can appear in an array. Cell arrays and pattern tables are specified by two dimensional color arrays. Color tables are specified by one dimensional color arrays. These basic value applies to a single dimension.
Default: 1024

Name: Maximum Point Array Length
Description: The basic value for the number of points and VDC that can appear in parameters for metafile elements.
Default: 1024

Name: Maximum String Array Length
Description: The basic value for the number of strings that can appear in an array. A data record is an array of strings.
Default: 1024

Name: Maximum String Length
Description: The basic value for the length of an individual string of characters.
Default: 256

The bundle representations are not settable in the current version of the CGM. This implementation dependency detracts from the open interchange of the CGM. The following default bundle table values will permit a picture to be uniformly rendered by all conforming interpreters.

Table 1: Default Bundle Tables

Bundle Type	Bundle Index				
Bundle Representation	1	2	3	4	5
Line Bundle					
Line type	Solid	Dash	Dot	Dash-dot	Dash-dot-dot
Line width	1	1	1	1	1
Line color	1	1	1	1	1
Marker Bundle					
Marker Type	Dot	Plus	Asterisk	Circle	Cross
Marker Size	1	1	1	1	1
Marker Color	1	1	1	1	1
Text Bundle					
Font Index	1	1			
Text Precision	Stroke	Stroke			
Character Expansion Factor	1	0.5			
Character Spacing	0	0			
Text Color	1	1			
Fill Area Bundle					
Interior Style	Hollow	Solid	Pattern	Hatch	Empty
Fill Color	1	1	1	1	1
Hatch Index	1	1	1	1	1
Pattern Index	1	1	1	1	1
Edge Bundle					
Edge Type	Solid	Dash	Dot	Dash-Dot	Dash-dot-dot
Edge Width	1	1	1	1	1
Edge Color	1	1	1	1	1

8 Registered Elements

8.1 Fonts

The fonts in Table 8.1 are public domain fonts, available from the National Bureau of Standards. Refer to the references in section 2. All of these fonts are considered to be basic capabilities of the TOP/MAP CGM Application Profile. Any of these fonts may appear in the Font List Element in a CGM that conforms to this application

profile. The font names are a concatenated string of "font vendor" + ":" + "font name".

Table 8.1: Basic Font Names

1. HERSHEY:CARTOGRAPHIC ROMAN
2. HERSHEY:CARTOGRAPHIC GREEK
3. HERSHEY:SIMPLEX ROMAN
4. HERSHEY:SIMPLEX GREEK
5. HERSHEY:SIMPLEX SCRIPT
6. HERSHEY:COMPLEX ROMAN
7. HERSHEY:COMPLEX GREEK
8. HERSHEY:COMPLEX SCRIPT
9. HERSHEY:COMPLEX ITALIC
10. HERSHEY:COMPLEX CYRILLIC
11. HERSHEY:DUPLEX ROMAN
12. HERSHEY:TRIPLEX ROMAN
13. HERSHEY:TRIPLEX ITALIC
14. HERSHEY:GOTHIC GERMAN
15. HERSHEY:GOTHIC ENGLISH
16. HERSHEY:GOTHIC ITALIAN

8.2 Generalized Drawing Primitive Elements

The following GDP Elements are considered common enough within the TOP/MAP computer graphics community to warrant registration.

8.2.1 Axis

Most graphs require axis lines and scales to indicate the orientation and value of the plotted data points. The most common type of scaled axis is easily produced by this GDP. The GDP draws any length line at any angle, divides it into one-inch segments, annotates the divisions with appropriate scale values, and labels the axis with a centered title. When both the X and Y axis are needed, the GDP can be specified separately for each one.

GDP IDENTIFIER - -1
NUMBER OF POINTS - 4
LIST OF POINTS - (x1) xcoor - X coordinate of the axis line's start point in VDC.
(y1) ycoor - Y coordinate of the axis line's start point in VDC.
(x2) divis - The division along the axis in VDC. If VDC Scaling Mode was metric with a scaling factor of 25.4, then to have 1 inch divisions this value would be 1.
(y2) axlen - Length of the axis line in VDC.
(x3) angle - Angle at which the axis is to be

drawn. Normally this is zero for a x-axis and 90.0 for an y-axis.
(y3) tickv - The value that will appear at the first tick mark on the axis.
(x4) tickd - The number of data units per division along the axis. This value is added to the "tickv" parameter for each succeeding division along the axis.
(y4) This value is ignored

GDP DATA RECORD - Single string of text. This is the title to be placed on the axis.

8.2.2 Grid

Some graphs require a grid to be overlaid on top of the plot data points. This GDP will permit specification of an overlay grid in a compact manner.

GDP IDENTIFIER - -2
NUMBER OF POINTS - 3
LIST OF POINTS - (x1) xcoor - X coordinate of the grid's start point.
(y1) ycoor - Y coordinate of the grid's start point.
(x2) xlen - Length of grid along the x-axis in VDC.
(y2) ylen - Length of grid along the y-axis in VDC.
(x3) xdelt - Delta between x-axis grid lines in VDC.
(y3) ydelt - Delta between y-axis grid lines in VDC.

GDP DATA RECORD - N/A

8.3 Escape Elements

The following Escape Elements are considered common enough practice or of important enough use to warrant registration.

8.3.1 Disable Clearing of Viewsurface

Normally, the viewsurface will be cleared on each Begin Picture Element. This Escape Element will disable the clearing of the viewsurface for the picture that follows this element. This Escape Element must precede every Begin Picture Element that corresponds to the picture that is to be overlayed on top of the current picture. This Escape Element will have no effect on the resetting of the metafile defaults on each Begin Picture Element. This Escape Element is a basic capability of the TOP/MAP CGM Application Profile.

ESCAPE IDENTIFIER = -1
ESCAPE DATA RECORD = N/A, ignored if specified

8.3.2 Dash Line/Edge Type

A significant number of computer graphics systems permit a client to program the line type for line and edge attributes. This Escape Element defines the dash and gap length for line and edge attribute type "dash". The Escape Element will only affect subsequently defined line or fill area graphical primitives. This Escape Element is a basic capability of the TOP/MAP CGM Application Profile.

ESCAPE IDENTIFIER = -2
ESCAPE DATA RECORD = A single string of text. This is the definition of the length of the line segment and the gap in VDC. The string is in a FORTRAN F9.4 format. There is no separator between the two format specifications.

For example, a dash line type with a line segment of 0.4 VDC and a gap segment of 0.25 VDC would be coded with the following Escape Element.

ESCAPE IDENTIFIER = -2
ESCAPE DATA RECORD = "0000.40000000.2500"

8.3.3 Dash-Dot Line/Edge Type

A significant number of computer graphics systems permit a client to program the line type for line and edge attributes. This Escape Element defines the dash, dot, and gap length for line and edge attribute type "dash-dot". The Escape Element will only effect subsequently defined line or fill area graphical primitives. This Escape Element is a basic capability of the TOP/MAP CGM Application Profile.

ESCAPE IDENTIFIER = -3
ESCAPE DATA RECORD = A single string of text. This is the definition of the length of the long and short line segment and the gap in VDC. The string is in a FORTRAN F9.4 format. There is no separator between the three format specifications.

For example, a dash-dot line type with a long line segment of 0.875 VDC, a short line segment of 0.5 VDC and a gap segment of 0.25 VDC would be coded with the following Escape Element.

ESCAPE IDENTIFIER - -3
ESCAPE DATA RECORD - "0000.87500000.50000000.2500"

8.3.4 Dash-Dot-Dot Line/Edge Type

A significant number of computer graphics systems permit a client to program the line type for line and edge attributes. This Escape Element defines the dash, dot, and gap length for line and edge attribute type "dash-dot-dot". The Escape Element will only effect subsequently defined line or fill area graphical primitives. This Escape Element is a basic capability of the TOP/MAP CGM Application Profile.

ESCAPE IDENTIFIER - -4
ESCAPE DATA RECORD - A single string of text. This is the definition of the length of the long and short line segment and the gap in VDC. The string is in a FORTRAN F9.4 format. There is no separator between the three format specifications.

For example, a dash-dot line type with a long line segment of 0.75 VDC, a short line segment of 0.375 VDC and a gap segment of 0.2 VDC would be coded with the following Escape Element.

ESCAPE IDENTIFIER - -4
ESCAPE DATA RECORD - "0000.75000000.37500000.2000"

Final Report

NBS Order Number 43NANB612433

Evaluation of European Graphics Validation Suite

30 July 1986

GSC Associates Inc.
13663 Prairie Avenue
Suite B
P.O. Box 2286
Hawthorne, CA 90251
213-978-9351

CONTENTS

1.0 Introduction

This document is the final report by GSC Associates Inc. for contract number 43NANB612433. Under this contract, GSC Associates provided assistance to NBS in its work for the Department of Defense Computer Aided Logistic Support (CALS) program. The purpose of this contract was to make recommendations to NBS to help accelerate and complete the ongoing evaluation of the European graphics validation suite, and to provide a recommendation for a DoD graphics validation approach.

This introductory section will provide a summary of results and a description of the approach taken in evaluating the European validation tests. Subsequent sections of the report address each major category of test in turn -- data structure tests, error tests, and operator interface tests -- describing the structure of the tests, the results obtained by executing them on a commercially available GKS implementation, and conclusions regarding their utility for DoD purposes. A section is devoted to an analysis of how well a few selected GKS requirements are tested by the European validation tests. Finally, recommendations are made regarding future work and the utility of the tests for DoD graphics validation purposes.

GSC Associates wishes to acknowledge the assistance of TRW Defense and Space Systems Group in completing this work. TRW assisted by making a VAX 8600 computer and Tektronix 4128 graphics terminal available to us for executing the test programs.

1.1 Statement of the problem

The great diversity of graphics packages with different philosophies has inhibited the development of graphical applications software. Graphics standards -- such as the Graphical Kernel System (GKS) and the Computer Graphics Metafile (CGM) -- have been developed to help eliminate this problem and to encourage portability between different environments. A validation procedure is necessary to insure that implementations of standards, such as GKS, are consistent with the standards. Without this consistency, the advantages of portability are lost.

Recognizing this, the European Community sponsored a series of workshops during 1981 and 1982 to develop a methodology for testing GKS implementations. The development of a suite of tests -- based on the methodology developed at these workshops -- was subsequently funded. The actual development work was done at the Technical University of Darmstadt in the Federal Republic of Germany, and the University of Leicester in England. The development work was supervised for the European Commission by the GMD (Gesellschaft fur Mathematik und

Datenverarbeitung) in the Federal Republic of Germany.

The European test suite is supposed to subject a completed GKS implementation to a thorough test of its consistency with the GKS standards with hopes of discovering errors. The less errors a particular implementation generates, the greater degree of standardization it contains, yet the lack of errors generated does not guarantee correctness. As the test suites are developed further, the degree of confidence in the correctness of the implementation will increase. Basically, the GKS implementation is tested by calling GKS functions, sometimes in conjunction with input devices, which generate a response. The responses from these calls are then evaluated and corresponding error messages reported if the responses are not as those designated by the GKS standard. The operator interface tests are evaluated a little differently since they require human visual evaluation of the graphical output from the device chosen for test.

The GKS validation test suite developed under the sponsorship of the European Community consists of three sets of tests. One set tests the data structures internal to GKS; a second set tests the errors that occur when executing GKS functions; and the third set tests the general functionality of GKS at the operator interface level. All of these tests are at the level of what would commonly be referred to as "system acceptance tests" in DoD terminology.

The data structure tests consist primarily of a dialog between the certification program and the GKS implementation. GKS routines are correctly called under valid states of GKS in order to determine the viability of the implementation. The responses of the GKS implementation are then written to a report file.

The error tests check the response of the GKS implementation to deliberately induced error situations. Specifically, the error messages returned by the implementation are compared with a list of correct responses, and reports of these comparisons are written to a report file.

The Operator Interface Tests consist of an "Operator Script" and output to the screen of a workstation. The Operator Script tells the operator what the screen should be displaying, and provides a form for noting the agreement or discrepancies between the scripted version and the actual content of the screen display.

The purpose of the contract reported on here was to advise the NBS concerning the suitability of basing a DoD validation procedure for GKS implementations on the European GKS validation suite. If these tests are of sufficient quality, their use can save the substantial development effort needed to implement alternative tests.

1.2 Approach to the problem

To evaluate these tests, GSC Associates devised a three part strategy. The first part of the strategy involved installing and executing the tests in a typical environment in the United States. The second part of the strategy involved analysing the structure of the tests themselves to determine the quality of their construction and their modularity and ease of use in testing GKS implementations, especially on smaller computers or in imbedded processors. The third part of the strategy involved a detailed investigation of the extent to which the routines test the requirements in the GKS specification. Each of these components of the evaluation strategy is discussed in more detail in subsequent paragraphs.

1.2.1 Installation and execution of tests

By installing and executing the European validation tests in a typical DoD contract development environment in the United States, a great deal can be told about their utility. GSC Associates has installed the validation tests from a distribution tape furnished by National Bureau of Standards on a DEC VAX 8600 running the VMS operating system. All three sets of tests have been executed against an ISSCO GKS implementation using a DEC VT240 terminal and the operator interface tests have been executed using a Tektronix 4128 terminal. In addition, some of the tests were executed using the Tektronix terminal and the DEC implementation of GKS.

The Validation Tests were developed in a European academic computing environment and have not seen extensive use within commercial production environments or the DoD contractor software development environment in the United States. By installing and executing the tests, several goals were achieved:

1. The effort required to install and customize the test routines for a new environment was evaluated.
2. The quality of tests was determined by scrutinizing test results to see if errors encountered during the execution of tests are due to errors in tests themselves or in errors in the GKS implementation.
3. The quality of the documentation furnished with the tests was evaluated. The test scripts furnished with the operator interface test in particular, were checked for ease of use and quality.

The results of executing individual sets of tests are contained in Paragraphs 2, 3 and 4 of this

document. The results of the installation procedure itself, are discussed below. In general, we were disappointed with the quality of the installation instructions, the ease of customization of the routines to a new environment, and the quality of programming practice used in their construction. Specific comments are:

1. The test programs contain many spelling errors. Additional documentation of some of these can be found in item 2 of section 2.2 of this report. Even output to the screen during the operator interface test often contains spelling errors! The presence of such a large number of errors is indicative of a lack of care and a lack of proper quality assurance procedures in their development.

2. Directions for modifying test routines is contained both in the written documentation accompanying the tests and in the source code itself. For example, Page 8 of the documentation furnished with the tests discusses customization of the routine UWKVAL. When one begins modifying the code for this routine, additional items which must be changed are encountered in comments contained in the code itself. Excerpts from the installation instructions and from the subroutine UWKVAL that illustrate this are contained in Appendix C.

3. Page 7 of the documentation, line number 4, instructs the installer to use the text editor to search for the phrase "C EFFECT". From the documentation it is not possible to ascertain how many spaces occur between the letter "C" and the word "EFFECT", thereby, making it impossible to use a text editor to search for the string.

4. On page 8 of the documentation, Section 3.2.2.2, subroutine UWKVAL is misspelled as UNKVAL.

5. The documentation regarding modifications required to the ERNAME is very unclear, for example, what is the "GDP identifier" and where is the source of the information regarding it.

6. Some documentation in the test programs is in German. Appendix C contains a portion of subroutine PDINIT which illustrates this.

7. The structure of, and comments contained in, the headers to the test programs are of uneven and inconsistent quality. Maintenance logs contained in many programs should have been removed before the routines were delivered. Appendix C contains listings from subroutines ECHEKR and ESTART which illustrate this. Almost all of the programs and subroutines have these problems.

8. During installation, the identical item must be modified in a number of subroutines. Examples are the number of workstations, the list of the identifiers of workstations and the name of the GKS implementation. Proper programming practice would be to use a single INCLUDE file to define all of these items in FORTRAN source code which is the automatically "included" in each program at compile time. This would increase the modularity of the tests and prevent inadvertent errors due to inconsistent updates of source code.

9. There are apparently no standards for naming items within test routines. For example, the number of workstations is called NUMWK in one routine and NWK in another. This makes it difficult to understand how the routines function when problems occur.

10. A number of special FORTRAN statements have been left in the test routines for the PERQ graphics system. Although these have been commented out, one wonders if this extensive level of customization is required for testing other GKS implementations. If so, general instructions should be given for adding these items, otherwise, they should be deleted from the delivered validation suite. Appendix C contains a portion of subroutine ESET which illustrates this.

11. The installation instructions contained in some of the programs allocate storage for the item BUFA. Nothing explains the use of BUFA or the appropriate value to be used for it. After some analysis, it appears that the length of this array corresponds to the number of memory units contained in the OPEN GKS function call.

12. Documentation contained in the OINIT routine states that it belongs to Version 2.0 of the test suite, however, written documentation furnished with the tests say that they are Version 1.0. Also, this procedure prints out lines to the screen which are cut off, for example, "GKS TEST SUITE..." becomes "KS TEST SUITE..." and "IMPLEMENTATION:" becomes "MPLEMENTATION:". The WRITE statement that prints out these lines needs a carriage control character as the first character in the front of the text as documented in listing of OINIT in Appendix C.

13. The last line of Page 10 of the documentation refers to "graphics file". It appears that this is a reference to a graphical metafile but it is confusing to installers to refer to it as a graphical file. This is a general problem with the names of other files used by the test programs since they are not descriptive enough of their general function and use. In particular, the names for error file are inconsistent and it is difficult to tell which file is used for which purpose, without making a

detailed analysis of the code. This can be seen in subroutines D2011 and UFUNS.

14. In general, no use was made of proper structured FORTRAN programming practices in writing the test routines. The INCLUDE statement - which could provide a large degree of modularity and ease of customization to the programs - does not appear to be used at all. Some critical items, for example, the number of supported workstations and the names of the supported workstations - are hard coded in subroutine D2011 ! Many common blocks are used throughout the test programs. They should be defined once and then INCLUDE'd in programs rather than hard coded into each routine.

15. Items in some common blocks appear to be initialized with assignment statements rather than through data statements. This is very poor programming practice.

16. In the code documentation, there are many grammatical errors, especially in the data structures tests. Appendix C contains the listing of program D2013 which illustrates this.

17. Before procedure ESTART could be compiled, the statement :

```
OPEN(UNIT=FUNIT,FILE=FILENAME(1:10))
```

which caused the information message:

```
%FORT-I-DEFSTAUNK,Default STATUS='UNKNOWN' used in OPEN statement
[LE=FNAME(1:10))] in module ESTART at line 90
```

was modified. This procedure now compiles without producing the information message after changing in this line to :

```
OPEN (UNIT=FUNIT,STATUS='NEW',FILE=FILENAME(1:10))
```

It is poor programming style to allow such a file OPEN statement to take a default status value since it can lead to inadvertent destruction of the opened file. The correction to this coding error is documented further in Appendix C.

18. The installation instructions do not contain a list of temporary files which are created during execution of the tests. There appears to be no consistent philosophy regarding the use and naming of temporary files either. The tester must inspect his directories at the conclusion of a

test and guess which files should be printed and/or deleted.

19. A list, included in the installation documentation, of which test programs use GKS segments would aid the user in knowing when to add code to allocate a common block of segment work space. The adding of this common block is a customization needed when dealing with segments in certain GKS environments. Since some GKS environments reserve their own segment work space without the user needing to add a common block allocating storage for it, this implementation-dependent customization can not be included in the test programs. It is something that should be noted in the documentation, however.

20. Many of the test programs have the following text hard-coded into them: CHAR*80 WKNAME (46) and PARAMETER (NAWK=46). It would be more effective to make these environment dependent parameters so they can be initialized only once.

21. Many of the programs and the subroutines they call could use some documentation giving a brief description of what they perform at the beginning of the code, for example, EREP, ERNAME, ERQPXR, ESET, ESEXER, and others have little or no internal documentation.

1.2.2 Analysis of test structure

All main programs of the data structure and error tests were inspected to determine which subroutines were used. This process was continued for subsequent calls to subroutines until the lowest level of detail was reached. This resulted in a complete picture of the hierarchical structure of the test suites, some of which is presented in the appropriate paragraph for each class of test below. The size of the load module for each main program for the VAX environment is given in Appendix A. The average size load module and the smallest size load module for each test are given in Table 1.

Class of Test	Average Size	Maximum Size
Data structure	410 K	546 K
Error handling	384 K	468 K
Operator interface	435 K	494 K

Table 1. Load module sizes

The load module sizes for GKS implementations in other environments are expected to be comparable to those in Table 1. Based on this information it appears unlikely that the tests will be usable in environments where there is less than 512 K bytes (half a megabyte) of available RAM.

1.2.3 Requirements traceability

A detailed examination of GKS requirements relating to the POLYLINE primitive was performed to analyze the test coverage for that portion of GKS. The methodology followed was:

1) The GKS Standard was examined to determine requirements relating to POLYLINE.
2) A subset of these requirements, relating specifically to transformations, and the attributes POLYLINE Index Linetype ASF, Linewidth Scale factor, Polyline Color Index, Linetype ASF, Linewidth Scale Factor ASF, and Polyline Color Index ASF was chosen. This subset was selected to examine these basic features in some depth.
3) The degree to which the validation tests tested these requirements was then determined.

This process is complicated by the fact that the GKS standard is not written in a form where requirements are clearly and explicitly stated. To illustrate how testable requirements were derived from the GKS standard, consider the following extract which defines the GKS Inquire Polyline Facilities function:

INQUIRE POLYLINE FACILITIES **GKOP,WSOP,WSAC,SGOP** **Lma**

Parameters:

In	workstation type			N
Out	error indicator			I
Out	number of available linetypes		(4..n)	I
Out	list of available linetypes		(-n..-1,1..n)	n×I
Out	number of available linewidths		(0..n)	I
Out	nominal linewidth	DC	>0	R
Out	range of linewidths (minimum,maximum)	DC	>0	2×R
Out	number of predefined polyline indices		(0,5..n)	I

Effect: If the inquired information is available, the error indicator is returned as 0 and values are returned in the output parameters.

If the number of available linewidths is returned as 0, the workstation supports a continuous range of linewidths.

If the inquired information is not available, the values returned in the output parameters are implementation dependent and the error indicator is set to one of the following error numbers to indicate the reason for non-availability:

8 GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC or SGOP
22 Specified workstation type is invalid
23 Specified workstation type does not exist
39 Specified workstation is neither of category OUTPUT nor of category OUTIN

A number of explicit requirements were derived from this text. One of the derived requirements and a description of what must be done to test it adequately are given below:

Requirement:

When the Inquire Polyline Facilities function is invoked and GKS is not in one of the states GKOP, WSOP, WSAC, or SGOP, then the error indicator must be set to 8.

Test:

Put GKS in state GKCL and call the Inquire Polyline Facilities function. Check that the integer value 8 (indicating error number 8) is returned in the error indicator parameter.

Based on derived requirements and test descriptions -- such as the one illustrated above -- the source code for the validation tests was inspected to determine the degree to which requirements were tested. The results of this evaluation are presented in Section 5 of this report.

1.3 Summary of results

The test suites were successfully installed on a Digital Equipment Corporation VAX 8600 computer running the VMS operating system. The graphics capabilities available on this computer include the ISSCO and the Digital Equipment Corporation (DEC) implementations of GKS. Graphics output devices included a Tektronix 4128 terminal and DEC VT240 terminals. A moderate amount of difficulty was encountered in installing the tests due to the poor quality of the documentation furnished with the test suite. For example, instructions detailing how to customize certain subroutines for a particular GKS implementation were contained in the written documentation accompanying the test suite and others in comments contained in the source code to the subroutines themselves. For several routines, both sets of instruction had to be followed.

The error tests and data structure tests were both completely executed using the ISSCO GKS implementation and with VT240 terminal. A large number of error messages were generated as a result of the execution of these tests. For the data structures tests, some of the errors were programming errors in the GKS test programs and some were errors in the ISSCO GKS implementation which were discovered by the tests. Problems encountered executing the error handling tests were due mostly to programming errors in the GKS test routines. The operator interface tests were also completely executed with more favorable results. A moderate number of errors were encountered executing these tests with both the DEC VT240 terminals and the Tektronix 4128 terminal and most were due to errors in the ISSCO GKS implementation.

A determination of the structure and organization of the test programs was completed. During inspection of source code for the routines, numerous problems with programming practices used in their construction, the quality of their internal documentation and FORTRAN language coding errors were encountered. These problems are documented below in the discussions of specific tests. Our overall impression is that the data structure and error tests are of very low quality. The operator interface test source code is better, but still contains a number of errors and examples of poor programming practices.

The degree to which the validation tests determine conformance to GKS requirements was evaluated. This was done by taking a representative set of requirements - - some of those associated with the POLYLINE output primitive of GKS - - and determining if these requirements are tested in the validation routines. The more obvious requirements were usually well covered by tests at an appropriate level for an acceptance test procedure. Less obvious requirements were poorly tested or not tested at all. Some requirements were identified that could not be tested through the GKS subroutine call interface. Other forms of requirements verification -- such as analysis or "internal unit-level" tests must be used to verify these requirements

Our conclusion is that considerable effort will be required to bring the European validation suites up to an acceptable level for either DoD purposes or for use in "commercial" testing in the U.S. Effort will be required to properly document the internal structure of the tests, rewrite portions of them consistent with good programming practices, correct coding errors, and add additional tests to increase the degree of requirements validation. A detailed list of recommendation is given in Section 6 of this report.

2.0 Data structure tests

The data structure tests check that the various GKS state lists are correctly updated as a result of calls to GKS functions. There are 30 data structure test programs which are all contained in the test directory VPROG. The following paragraphs describe the structure, results of execution, and our conclusions regarding these tests.

2.1 Structure and philosophy of the tests

These thirty data structure tests each have a name which begins with a character "D" which denotes that they are data structure tests. A common set of subroutines are used in each test. Routines which are called in most of the tests are PDINIT, PDTRNP, CWRITE, DINIT, CPUT, and CNUMER. These routines perform the following functions:

PDINIT - initializes the description file XAIDSR.CIP if necessary, initializes the common areas, and requests the current test level and whether the trace facility is desired from the tester;

PDTRNP - writes a program or procedure name to the trace file;

CWRITE - writes a program name and error message to an error report file;

DINIT - opens GKS and then opens and activates the requested number of workstations;

CPUT - puts a program name and message number onto a message stack to later be used in an error report file;

CNUMER - writes the number of detected errors, test level, and supported workstations to the screen and error report file.

For the most part, other subroutines in the data structure tests are used to call specific GKS functions. For example, DOPKS calls the OPEN GKS function -- GOPKS. These subroutines are named by replacing the "G" sentinel character in the FORTRAN Language Binding of a GKS routine with the character "D".

To illustrate the degree of care exercised by the writers of the data structure tests, test program D2012 was analyzed in detail. The following paragraphs lists specific problems found in this test program and its documentation. A complete listing of this program is contained in Appendix D. This listing is annotated with numbers keyed to the following comments.

1. The documentation in the header for this program states that it is a "LOA" Test Program rather than a "L0A" Test Program. This is a typographical error resulting from using the capital letter "O" rather than the numeral "0".

2. In describing the effect of the test program, the word INDEPENDENT is misspelled INDEPENDED.

3. In the describing the errors generated by this program, Error 1390 is described as "Try to Set Linewidth Scale Vector" rather than "Try to Set Linewidth Scale Factor". The proper term (according to the GKS standard) is Linewidth Scale Factor.

4. Spaces are omitted from comments many places in the documentation. For example, "creation date", "trace mode", and "logical unit number" are all printed without one or more of the appropriate spaces.

5. Some internal documentation remains in the headers of the test programs and was not removed prior to distribution. For example, the project is identified as "VALGKS Application" and the update history of the program is given. These are not of interest to anyone but the original developer of the program.

6. The header of the test program indicates that it tests "GKS 7.4". This refers to a early DIN version of GKS rather than to the current International Standard version. The documentation should state that what is being tested is ISO GKS (ISO 7942), not GKS 7.4.

7. An inconsistent style of continuation statement is used in the declaration of common block PDCTRL. On some lines, the comma indicating the separation between adjacent items in the common block is included at the end of a line and on some lines it is carried over to the beginning of the line.

8. The comment statement, "Dummies For Not Used Parameters" is grammatically incorrect.

9. In the comment for "Intern Integer Variables", the word internal is abbreviated unnecessarily to "intern" rather than the full word "internal". Such stylistic conventions make the internal documentation of the tests more difficult to read.

10. Several large common blocks which occur in many of the data structure test programs are reproduced in the code of each program rather than being included from a common external file.

11. Both the asterisk (*), plus sign (+), and numeral 1 are used in column 6 to indicate continuations. This is poor programming style.

12. The program documentation makes no distinction between the notion of <u>available</u> workstation and <u>active</u> workstation. For example, the parameter NAVWK is declared to represent the number of <u>available</u> workstations but when it is used as a limit for the DO loop ending at line 100, it is stated to represent the number of <u>active</u> workstations.

13. The style in which the FORTRAN code is written is extremely poor and shows a total disregard for consistent programming practice. For example, indentation is used inconsistently to set apart portions of the code which are inside IF statements and DO loops. The listing in the appendix is annotated to show two IF statements, one of which is set apart by a single space of indentation and the other one of which is not indented at all. It also shows a DO loop which is not set apart by any indentation.

14. The main portion of the test program loops over all active workstations and attempts to set a range of values of indices associated with Polyline attributes. The listing in Appendix D is annotated to show one such call to subroutine DSPLI. The listing states that the purpose of this call is to "Set Minimal Index". When the code for subroutine DSPLI is inspected, one quickly discovers that in fact, DSPLI performs many more functions than simply setting a Polyline Index. In fact it appears to set values for many other Polyline attributes. This causes severe problems when errors occur in the test since a routine may generate a large number of unexpected errors and there is no indication in the higher level documentation that these errors could be generated. A listing of routine DSPLI is included in Appendix D to illustrate this fact.

15. When an index -- such as Polyline Index -- which can only take values from a small discrete set is tested it would be most appropriate to test all the values than simply testing the minimal one, maximal one, and the value in between. This would be easily accomplished, would take very little additional execution time, and would provide much higher degree of confidence in the integrity of the GKS implementation. The strategy of testing minimal values, maximal values, and a value in between is most appropriate for those item which take their values from a continuous range.

    The documentation in the headers states that no random values are used for parameters. It would be better however, to generate a random value, or even better, a set of random values between minimum and maximum values for certain parameters for testing purposes, instead of using a point in the mid range.

16. During the inspection of the codes for subroutines DSPLI and DSPLCI, it was discovered that they were virtually identical. This is the case with almost all of the data structure tests. A large number of routines - some consisting of hundreds of lines of code - are identical except for a few lines. Appendix D contains listings of routines DSPLI and DSPLCI both with annotation showing the differences in the programs. Many of these programs could be collapsed into a single routine with a single parameter passed as an argument indicating which alternative lines should be executed. It would appear that the developers of these test programs were paid based upon the number of lines of code generated rather than the quality of the code.

17. In the next to the last line of the program, the word program is misspelled PROGRAMM.

18. There is no list provided of all of the errors generated by the data structure test. If for example, error number 1367 occurs, and the tester wants to know what this error is and which routines could generate it, the only way to determine this is by reading the headers for all the individual test programs!

2.2 Results of executing tests

A number of errors in the code and documentation of the test programs were encountered during the process of executing the tests. In addition, several errors in the ISSCO GKS implementation were found as a byproduct of executing the tests. A representative sampling of both are explained in this section.

2.2.1 General problems

1. The data structure tests are not modular. There is no apparent way to tell from the name of the program what is tested. Since many routines call GKS functions that are only found in higher level implementations, they couldn't even be linked to test the DEC level 0B implementation. The tests query the operator for the GKS level number, but this is of no avail if the proper subroutines aren't available for linking. Programming practice based on proper use of conditional compilation could avoid this problem.

2. There are many spelling errors displayed on the screen which include: "SPECIFIC" is misspelled as "SPEZIFIC" and "WORKSTATION INDEPENDENT" is misspelled as "INDEPENDEND" during the queries for the creation of the file XAIDSR.CIP; "WANT TRACE PROTOCOL?" is misspelled as "PROTOCOLL" which is queried at the beginning of each test program; at the end of a run, where the programs display results, "CATEGORY" is misspelled as "CATEGORIE" and "PROGRAM" is misspelled as "PROGRAMM".

3. Much of the information queried for in the process of creating the file XAIDSR.CIP is already supplied in the device dependent routines after the user has changed these routines to work with the site's particular system. It is redundant to query for this information.

4. In the creation of the file XAIDSR.CIP, the program at one point queries for information regarding the OUTIN workstations to be tested. When it asks for the connection type for the workstation, it asks for the type for an INPUT workstation.

5. When the file XAIDSR.CIP is being created, the user is occasionally queried for numerical data and is prompted with the line, "FORMAT=I2". On the next line, "12" is then displayed. It is not clear what this is to designate.

6. At least 3 of these programs when executed do not return control to the tester and have to be aborted.

7. In data structures tests involving segmentation, at run time the error message ">>>> Error following link QGMSEG" and also ">>>> Improper medium code 0; LSU0" are displayed on the screen. It is not clear as to what is causing this problem. In program D2013 these messages seem to be generated after the call to GCRSG (Create Segment) within the subroutine DCRSG. Other test programs which generate these messages are: D2016, D2026, D2036, and

D2056. These error message do not appear to be generated by the validation tests and appear to be due to errors in the ISSCO GKS implementation. It is interesting to note that the tests do not explicitly detect these errors.

8. In several programs, in particular D2024, errors are generated regarding the manipulation of tables dealing with Pattern fill area style. Pattern is not supported on the Tektronix 4128 nor the VT240 and in several tests the code should query about Pattern facilities before trying to test them. Other programs affected by this problem are: D2041 and D2042. Program D2024 and its error messages regarding this problem are documented in Appendix D. Program D2024 needs a call to GKS function GQPAF (Inquire Pattern Facilities) in the code before an attempt to use Pattern is made. This is not done in any of the mentioned programs. Based on the results of this inquiry, use of Pattern should not be attempted if it is not supported. It appears that the programs designed to test implementations of the device-independent GKS standard are not them themselves device-independent!

2.2.2 Problems in specific tests

Test D2011. A "current GKS language binding error" number 2002 is reported when executing this test. This error indicates that an attempt was made to access a non-available element of a set or list. It occurs in the call to GQACWK (Inquire Set of Active Workstations) and is caused by the test program calling GQACWK with the Set Member Requested parameter greater than the implementation's "Number of Active Workstations." Similar errors occur in many of the data structure tests and appear to be due to the "hard-coded" parameters in the source code and the failure to code in a device- and implementation-independent manner by inquiring certain facilities before attempting to access unsupported features or elements.

Many times the error messages generated by test programs do not indicate specifically what or where the error is. For example one error generated in D2011, is "1350 Program starts with errors". No information is provided to tell the tester what problem exists or how to correct it. The code generating the error and the error message from D2011 are documented in Appendix D.

Test D2021. There is an access violation error from line 166 which calls DOPWK in which line 393 holds the error. The run time error received was:

%SYSTEM-F-ACCVIO, access violation

DOPWK is called and within that there is a call to GQSKS (Inquire stroke device state) which is the line with the error. Insufficient time was available to completely investigate the cause of this error, but it appears to be due to an error in the ISSCO GKS implementation.

Test D2024. An error message regarding GKS function GQCR -- color representation not delivered -- is generated when run on the VT240 (which is a monochrome device that doesn't support color)but is not generated on the Tektronix 4128. The VT240 also reports GKS error number 94 ("A representation for the specified color index has not been defined on this workstation".) This is not an error in the implementation but rather in the test program. The test program should inquire a workstation about its color facilities using GQCF (Inquire color facilities) before issuing this GQCR inquiry on a workstation that doesn't support color. The code and error messages regarding this problem are documented in Appendix D.

Test D2041. There was a "current GKS language binding error number 2002" involving a call to GQACWK(Inquire set of active workstations). After inspecting the code for this test it was determined that this message was due to the ISSCO GKS implementation passing back an invalid error indication to the test program in the error indicator parameter. For this particular call, the only valid error indicator values are 0 and 8. The test did not explicitly find this error and actually made it difficult to diagnose due to the cryptic diagnostic message provided to the tester. An explicit message such as "GQACWK returned an invalid error indicator value of 2002. The valid values are: 2, 8" would aid in diagnosing such problems. The code and the error message are documented in Appendix D.

2.3 Conclusions

The internal documentation for the data structure test describes the overall effect of the test. The programming style and degree of commenting does not permit a quick reading of <u>how</u> the effect is achieved. This is a definite deficiency from a maintenance point of view.

The whole philosophy behind the data structure tests is flawed. Since no graphical output is produced during the tests, an implementation could pass by correctly implementing the update of certain internal data items in response to the invocation of GKS functions. Unfortunately, updating the data structures alone is not sufficient. An implementation must modify its future behavior -- especially the graphical output it produces -- based on these values. <u>This is not tested</u>! The only valid way to test the GKS data structures is to interleave such tests with the production of graphical output. Demonstrating that a value in a state list can be set is of little value if the implementation makes improper use of that value.

Execution of the tests provides only a very minimal amount of information regarding the correctness of a GKS implementation. Many of the test routines perform functions other than those expected and many contain nearly identical code. The diagnostic messages provided are cryptic and nearly useless in locating and correcting defects in an implementation. The quality of their source code and documentation is extremely low.

3.0 Error tests

The purpose of the error tests is to determine if error conditions are properly raised as the result of executing GKS functions. The error tests are contained in test directory VERR and consist of 28 test programs and a number of subroutines. A description of the structure of the tests, the results obtained in their execution, and conclusions regarding their quality are contained in the following paragraphs.

3.1 Structure and philosophy of the tests

Twenty-eight main test programs are provided. The naming convention for these programs is as follows. Each program begins with the letters "ER" followed by two characters indicating the lowest level of GKS to which the functions being tested belong, followed by two digits indicating the number of this test within that level. For example, the first test program is named ER0A01, indicating it is an error test which checks level 0A functions of GKS and it is the first test program in the sequence of level 0A error tests.

The order of the error test follows the organization of the GKS specification. For example, the first few level 0A tests test precisely the functions contained in sections of Paragraph 5 of the GKS Specification:

1) ER0A01 checks errors generated by control functions (Paragraph 5.2 of the GKS Specification);

2) ER0A02 checks errors generated by output functions (Paragraph 5.3 of the GKS Specification);

3) ER0A03 checks errors generated by output attribute functions (Paragraph 5.4 of the GKS Specification);

etc.

Within each test program there is a well defined structure of subroutine calls. Subroutines ETITLE, ESET, ESEXER, ECHEKR, ERNAME and EREP are used repeatedly in all of the error tests. These routines perform the following functions:

ETITLE - writes the title of the test to the report file;

ERNAME - determines the valid range of a name;

ESET - checks and sets the GKS operating state; sets the size of the buffer, and the number of supported workstations via implementation-dependent code supplied/modified by the tester; assigns a workstation identifier to the specified workstation and disconnects the window (for PERQ workstation only!);

ESEXER - sets up a list of expected errors;

ECHEKR - compares the errors that occurred with those that were expected;

EREP - writes a report of the errors to the report file.

A number of subroutines are used to call the GKS functions themselves and to generate error conditions. These are named by replacing the "G" sentinel character in the FORTRAN Language Binding of a GKS routine with the sentinel character "E". For example, the error test subroutine EOPKS, tests the OPEN GKS function - GOPKS.

3.2 Results of executing tests

During the process of installing and executing the error tests a number of errors in the tests themselves were found. In addition, one error in the ISSCO GKS implementation was discovered. A representative sample of these errors are documented in this section.

3.2.1 General problems

1. At least 2 error handling test programs do not return control to the tester when executed and had to be aborted.

2. Some of the text displayed to the screen is cut off, for example, during execution of the tests, the statement "EST IS RUNNING" is displayed instead of "TEST IS RUNNING". This occurs in subroutine ESTART and therefore happens during all error handling tests. This problem is caused by a FORTRAN coding error. The line of code which writes this to the screen does not include a carriage control character before the text "TEST IS RUNNING". The source code containing this error is documented in Appendix E. It is difficult to imagine such

an obvious error escaping detection and being delivered in the finished test routines. Perhaps they were developed using a FORTRAN compiler that was itself incorrect!

3. Many errors were generated during the execution of the error handling tests. In general, the only tests with few or no errors were the tests of inquiry functions. This is because these are the only tests to set the variable REP(I) in procedure ECHEKR which holds the reported error numbers. If the expected error number array EXPR(I) is not equal to the reported error number array REP(I) in procedure ECHEKR, the test fails. In the non-inquiry tests, REP(I) is never set and therefore holds the value zero. The only tests which are reported as passing in the non-inquiry tests are the ones with no expected error number (EXPR(I)=0). In addition, these non-inquiry tests all get the error message "GKS not in proper state" after each call to a GKS function since the error is generated by the test routine. The non-inquiry tests which are affected by this problem are: ER0A01, ER0A02, ER0A03, ER0A04, ER0B01, ER0B04, ER0B05, ER1A03, ER1A04, ER1A05, ER1A08, ER1B01, and ER2A01. This is a coding error in the tests which should be corrected.

   Two lines of code were added to subroutine ECHEKR which print the values of EXPR(I) and REP(I) to the screen. When non-inquiry programs ER0A03 and ER1A05 were run with these lines, the output showed that EXPR(I) always contained the expected error and REP(I) always contained zero. When the inquiry test ER0A05 was run with these lines, the values of EXPR(I) and REP(I) were always equivalent which caused the tests to pass. The lines added to ECHEKR are documented in Appendix E.

4. The philosophy followed by most of the tests in examining the reported versus expected errors from a GKS implementation is fundamentally incorrect. The tests set up a list of expected errors, call a GKS function under conditions where these errors <u>could</u> be reported, and then check that <u>all</u> the expected errors are reported. This approach can't work since the GKS standard places no requirements on exactly <u>which</u> errors are reported in situations where more than one error may occur. The only requirement is that <u>at least one</u> of the possible errors be reported.

3.2.2 Problems in specific tests

Test ER0A02. There is an error generated at line 42 calling subroutine ECA which has the error at line 102. The error message received at run time was:

```
%FOR-F- ADJARRDIM, adjustable array dimension error
```

A zero values in the parameters DIMX and DIMY (dimensions of color index array) in the call to GKS function GCA -- Cell Array -- cause this error since these parameters are required to be integers in the range [1..n]. This error is documented in Appendix E.

Test **ER0A03.** Calls subroutines ESTXCI and ESFACI which are not included in the Appendix of the GKS validation routines documentation.

Test **ER0A10.** Calls subroutine EQPFAR. It is not contained in the listing of subroutines furnished in the Appendix of the GKS validation routines documentation. Subroutine EQFAR is contained in the listing of subroutines furnished in the documentation for test program ER0A10. It is not called and no source code is provided. Apparently, subroutine EQPFAR was misspelled as EQFAR.

Test **ER0A12.** There is a run time error at line 32 calling subroutine EQPXA which has the error on line 75. The error message received was:

%SYSTEM-F-ACCVIO, access violation

1. The problem is that an incorrect FORTRAN binding for the call to GQPXA (Inquire pixel array) was used in the test suite. Code calling the correct binding was contained in the source listing but was commented out. Apparently, the test suite was modified to show that an incorrect implementation tested correctly and these modification were inadvertently left in the delivered code. One wonders if this is common practice in "validating" GKS implementations in Europe! There are 7 instances of this in the tests.

2. Once this error was fixed, an error on line 32 in the call to EQPXA which has the error on line 214. The error message received was:

%FOR-F-ADJARRDIM, adjustable array dimension error

This error is caused in the last call to GQPXA which has values of 0 and -1 for the parameters DX and DY (dimensions of color index array) while the GKS standard requires the values be positive integers. These coding errors and the changes are documented in Appendix E.

Test ER0B01. There is a run time error at line 42 calling subroutine EINSK which has the error on line 109. The error message received was:

%SYSTEM-F-ACCVIO, access violation

This error is caused by an incorrect FORTRAN binding call for GINSK (Initialize stroke) in the test program. The current GKS standard is:

```
CALL GINSK (WKID,SKDNR,TNR,N,IPX,IPY,PET,XMIN,XMAX,YMIN,YMAX,
        BUFLEN,INIPOS,LDR,DATREC)
```

This includes a parameter not included in the test program. The parameter is INIPOS which in the ISSCO implementation designates the following the "editing position". This error is documented in Appendix E.

Test ER2A01. Does not call subroutine GERHND even though it is listed in the directory and source code for it is included. Also this program calls subroutine GCEVTM which is not a GKS function. Code for this subroutine is not included anywhere in the test subroutine directories.

3.3 Conclusions

The error handling test programs have a well defined structure of subroutine calls and the internal documentation is a bit better than the data structures tests. The simplicity of the error routines makes the code almost self-documenting without extensive commenting. Many errors are reported when they are executed. Most of these are due to errors in the test programs themselves rather than in the implementation being tested. Several fundamental design flaws in the tests prevent them from providing much useful information or from being properly evaluated in our study. If time had been available to recode routines ESEXER and ECHEKR so that they worked correctly, a better evaluation of the error test could have been accomplished.

4.0 Operator interface tests

Perhaps the most useful portion of the European validation tests for GKS implementations are the operator interface tests. An operator at a graphics workstation executes a test script and verifies the functioning of GKS by the appearance of output on the screen and the actions that take place as a result of operator input. The operator interface tests consist of 25 main programs, each of which calls several subroutines.

4.1 Structure and philosophy of the tests

Twenty-five main test programs are provided. The "OP" at the beginning of each name denotes that these are operator interface tests. The next two characters in the name indicate the lowest level of GKS to which the functions being tested belong then, the next two digits denote the number of the test within that level. Subroutines used repeatedly in all the operator interface tests include: OINIT, UFUNS, OPENKS, OTESTR, and UWKVAL.

OINIT - queries for workstation to be used and returns special data needed for certain tests.

UFUNS - returns unit numbers and report file names.

OPENKS - opens GKS with buffer size provided by the tester.

OTESTR - checks if there is another VDU in addition to the graphics screen and outputs messages to a separate screen (if provided) or to the upper right part of the same screen.

UWKVAL - returns workstation id from table of connection id and workstation types provided by the tester.

For the operator interface tests, an operator script, of necessity, provides a detailed description of the graphical output that should be generated. The graphical output is pictorial in nature, and the comparison - by the operator - of what should be displayed versus what is actually displayed, consists largely of a visual comparison between a drawing in the manual and the display screen. Text output in this phase of the testing is also somewhat imprecise - the manual states " ... the output of a text string may be exceeded (sic) the unit square display surface. The pictures in the document should be used for guidance only."

Some improvement could be made in choosing graphical output characteristics that must be

perceived and differentiated by an operator (tester.) For example, the standard test pattern colors -- red, green, blue, cyan, magenta, yellow, white, and black together with a grey scale -- would be more objective than the "seagreen" and "burgundy" of the test programs.

4.2 Results of executing tests

A number of errors in the ISSCO GKS implementation and programming errors in the tests were located in executing these programs -- in particular, errors regarding output to the screen. A few documentation errors were also found. A representative set of these errors are documented in this section.

4.2.1 General problems

1. There were problems getting the terminal to recognize valid input responses for most of the input tests. This appears to be due to errors in the ISSCO GKS implementation that the tests correctly discovered.

2. When the operator interface tests are run using the VT240 using ISSCO GKS, many of the tests don't run properly and some of the ones that do only display parts of the images to be displayed. For example, all of the OP1Axx tests will display the title frame and menu and when a choice is selected from the menu the operator is returned to the menu without executing the choice. This appears to be due to errors in the ISSCO GKS implementation that the tests correctly discovered.

3. Regarding the fill area interior style HATCH, the integer values describing the style index in the ISSCO implementation are negative values (-3,-2,-1). These are proper values for implementation-dependent hatch styles. Since no hatch styles have been registered to date, the use of such implementation-dependent values is required if hatch style is supported by an implementation. The test routines test style index with positive values only and therefore the style index never gets set properly. This causes the image not show up - not even an outline of the area to be hatched. This causes problems in several test programs:
   a) OP0A07 - at checkpoint 2.5, the semi-elliptical shape is not displayed. Also, at checkpoints 3.4 and 4.6 there are no hatch examples displayed.
   b) OP1A04 - at checkpoint 2.3 there is no image under bundled or individual and at checkpoint 2.4 there is no image for individual. At frame 2, there are no images under the text "INSTY". At checkpoints 5.1d, 6.1d, and 7.1e, there are no mountains displayed.
   c) OP1A05 - at frame 4, the butterfly body is drawn yet the hatched wings are not. At

frame 5, no polygons are drawn at all. The error actually occurs in subroutine OBUFLY.

d) OP1A06 - in all frames of this test, the butterfly wings are not displayed. The error actually occurs in subroutine OBUFLY.

These problems are due to errors in the test routines and are further documented in Appendix F.

4.2.2 Problems in specific tests

Tests OP0A01 and OP0A02. There are a sequence of calls involving activating, deactivating, closing, and then reopening the same workstation. These two programs work fine on the VT240 terminal yet on the Tektronix 4128 terminal, the frames following the title frame are not displayed. A sample program was written which isolated the process of activating, deactivating, closing then reopening the Tektronix 4128 workstation. In these operator interface tests and the sample program, the second GOPWK (Open Workstation) call got an "internal error in GERLOG" error message and the rest of the calls following GOPWK had the error message "GKS not in proper state".

When the GDAWK (Deactivate Workstation), GCLWK (Close Workstation), GOPWK (Open Workstation), and GACWK (Activate Workstation) sequence is commented out the programs work fine. This problem is due to an error in the ISSCO GKS implementation -- apparently in the device driver for the Tektronix 4128 -- that was caught by the test routines. Unfortunately, due to the nature of the error it took a considerable time to determine what was causing the problem. The test routines do not adequately test valid state transitions. Although they uncovered this error, they provided no assistance in determining what caused it. This problem and its error files are documented in Appendix F.

Test OP0A04. At checkpoint 3.1, text on the screen reads "RANGE OF LINETYPES" and not "FULL RANGE OF LINETYPES" as designated by the documentation. This error is due to the test script accompanying the test suite and in the tests themselves not being consistent.

Test OP0A05. The first column displayed on the screen in frame 2 looks like it is all dots instead of the individual polymarkers. Also, the larger squares in the top row do not contain the dot polymarker inside. This is due to an error in ISSCO's implementation of GKS which the test correctly detected.

Test OP0A07. A run time error occurs on line 698. The error message received was:

%FOR-E-OUTCONERR, output conversion error

This error is displayed 5 times. This problem was caused by a FORTRAN coding error in the test program. The test program attempts to print the value of variable STYLID which is negative value. The WRITE statement used cannot print negative integers. Appendix F contains a listing of the erroneous code.

Test OP0A08. Cell arrays are not displayed at all. This is an ISSCO GKS implementation problem in which cell array only works sometimes.

Test OP0A10. After the first fish image is drawn and return is hit, nothing else is displayed on the screen when this test is run on the Tektronix 4128 terminal. This is because of the same problem in the ISSCO implementation described above for OP0A01 and OP0A02 regarding closing and reopening workstations. The test runs correctly on the VT240 terminal. The lines causing the trouble are documented in Appendix F.

Test OP0A11. At checkpoint 4 the red and green checkered flag at the top of the ship is not displayed. This is an ISSCO GKS implementation problem in which cell array is not regenerated dynamically. At checkpoint 7.1b on the Tektronix 4128, the display is not cleared and the image drawn in the lower left quarter has the text "GKS" so large that it has been clipped. This works fine on the VT240 terminal. This also appears to be due to a problem in the ISSCO implementation that was correctly discovered by the test program.

Test OP1A02. This test contains exactly the same code as test OP0A02 ! This appears to be an error in the construction of the source code for the test suite. The code for OP0A02 was probably inadvertently copied over that for OP1A02.

Test OP1A03. A run time error occurs on lines 754, 783 and 812. The error received is:

%FOR-E-OUTCONERR, output conversion error

The value of the variable FONT contains a negative integer value which cannot be directly printed to the screen. The test routines are strangely inconsistent in this area. Some handle this "problem" correctly by taking ABS(FONT) and printing the minus sign using GTX (Text). This programming trick is not used consistently where needed and therefore needless errors

occur. Such problems are due to the lack of application of formal software development practices such as design and code reviews. The erroneous lines and negative number output code are documented in Appendix F.

Test **OP1A04**. A message from the test appears which states that the maximum length of the fill area or pattern tables is less than 10 (min. required = 10) even though pattern is not supported on the VT240 or Tektronix 4128 workstations. This is an error within the test program.

1. At line 944, the following run time error occurs:

   %FOR-E-OUTCONERR, output conversion error

This is the same problem as described in OP1A03 where a negative value needs to be output in a special manner and this is not done.

2. At frame 4, after the Pac-man and car images displayed - if return is hit, the program leaves and a run time error is given at line 1327. Frame 5 also gets this error at line 1528. The error message received for both of these is:

   %FOR-F-ADJARRDIM, adjustable array dimension error

Pattern is not supported on the VT240 nor the Tektronix 4128 workstations and the program does not include this portion of the test within the last scope of the IF statement which queries to see if Pattern is supported. This program also does not query for supported pattern indices which would be another way to avoid this problem. So when these calls to GSPAR (Set Pattern Representation) are called, the values passed in the parameters MMX and NMX are 0 and this causes an adjustable array dimension error. These errors are documented in Appendix F. Once again, it appears that the test programs are not device-independent enough.

Test **OP1A05**. At frame 1, the flag in the first ship is not redrawn. This is an ISSCO GKS implementation problem where cell array is not regenerated dynamically when all segments are redrawn on the workstation.

In frame 5 the segment priorities are not changed so the reversed order of the polygons is not drawn. Within procedure HORSEH, the data for the polylines is scaled to fit into a given area. A FORTRAN programming error was found in this test. The variable SCALEF is not declared in this procedure nor is it initialized. Therefore it defaults to local variable status with the value

zero. This causes no horse head image to appear on the screen. This variable needs to be declared in the procedure and initialized to 1. This problem affects frames 1 and 4 of this test. Also as noted previously, some of the output to the screen is not displayed because of invalid hatch styles in the code from subroutine OBUFLY. These errors are documented in Appendix F.

Test OP1A06. At frame 2, the right side of screen doesn't get clipped like it is supposed to. This problem is due to an error in the ISSCO implementation that leads to an incorrect treatment of the segment transform in relation to clipping when segments are implicitly regenerated. Also as noted previously, some of the output to the screen is not displayed because of invalid hatch styles in the code from subroutine OBUFLY.

Test OP1A07. After the title is displayed and return is hit, only the Tektronix 4128 screen cleared; then the program disappears and doesn't return. This is the same problem as described above for the OP0A01 and OP0A02 tests regarding the closing and reopening of workstations. The lines of code causing this problem are documented in Appendix F.

Test OP1B01. At checkpoint 1.1 the screen also says "(BASIC REQUIREMENT)" but this is not shown in the documentation. This program doesn't respond to the pick input. This is due to errors in the ISSCO implementation.

At frame 2, the title, the circle, part of the boat and the stars are drawn then the program exits prematurely. This is also due to errors in the ISSCO implementation .

In the documentation at checkpoint 2.1, this program is designated as program OP0B04 instead of OP1B01.

4.3 Conclusions

The internal documentation in the tests is very thorough. A number of relatively minor coding errors were found in the test programs. Others probably exist that weren't detected. The tests were much more effective than the data structure and error tests at uncovering problems in the GKS implementation being tested. These tests appear to be of some utility in validating GKS implementations.

5.0 Requirements traceability analysis

The quality of any validation test is determined by how well it tests requirements of the system being evaluated. This is especially difficult to determine in the GKS environment since the GKS specification itself is not as well organized as a traditional DoD System Requirement Specification. Nonetheless, we undertook evaluation the test routines by picking one area of GKS - - the processing associated with producing the Polyline output primitive - - for extensive evaluation of requirements traceability. A partial list of GKS requirements relating to Polyline was developed and is contained in Appendix B. Due to the extremely large number of Polyline requirements that were found, we were only able to evaluate the testing of a subset of them.

Once these requirements were extracted from the GKS Specification, they were used to derive a minimal testable set of requirements to validate that a GKS implementation conforms to the requirements for processing Polylines. The three sets of GKS tests were evaluated to determine how well they test each of the derived requirements. From this exercise, we can infer the degree of care used to construct GKS tests programs, the percentage of coverage of GKS requirements which they are likely to provide in an acceptance test situation, and the general quality of the tests.

5.1 Conclusions

The detailed results of the evaluation are presented in Appendix B. In this Appendix, a representative set of requirements is listed, together with a description of how the requirement should be tested, an identification of one or more places in the validation suite where it is tested, and a determination of how well it is tested.

The degree of requirements coverage in the European validation suite is reasonable for an acceptance test situation, especially considering the lack of organization of the GKS standard and the difficulty of extracting testable requirements from it. All key features of GKS were tested in one or more places. However, we easily uncovered meaningful requirements which were not adequately tested and found requirements that could not be tested at the GKS language binding interface.

If additional confidence in the correctness of a GKS implementation is needed, then additional requirements verification must be performed. This could be done by analysis of the source code of the implementation or by demonstrating the correct performance of a set of "internal unit-level" tests designed to check the correct implementation of key features -- such as transformations and certain

approximations -- that aren't visible enough through the subroutine call interface to be adequately testable in a validation test suite.

6.0 Recommendations

Based on our extensive analysis of the European graphics validation suite and our experience installing and executing the tests in a typical US graphics environment, we make the following recommendations to NBS:

1. The data structure tests are of too low quality to be useful for validating GKS implementations for DoD purposes. Rather than expending the resources necessary to correct deficiencies in the tests, and due to the fundamental flaws described in Section 2.3 above, we recommend that the operator interface tests be expanded to include testing of the most important data structures.

2. The error tests are of higher quality than the data structure tests. If the fundamental flaws discussed in Section 3.3 above are corrected, the tests could be used to provide a useful validation of the error handling of GKS implementations. A moderate effort would be required to upgrade the routines. The error tests should remain a separate set of tests since they are difficult to integrate with the operator interface tests.

3. The operator interface tests are of definite value in validating GKS implementations for DoD use. A moderate effort would be required to correct programming errors in the tests and to make them more device-independent. As stated in (1) above, the test could be expanded to include tests of the most important GKS data structure with little impact on their run-time efficiency.

4. The test programs are available only in FORTRAN. Since DoD environments are likely to use Ada™ consideration should be given to converting the tests to that language. This would be a fairly straightforward, but time-consuming process due to the simple structure of most of the test programs and subroutines.

5. Consideration should be given to restricting some of the options available in GKS when it is used in a DoD environment. Additional constraints should also be placed on the "correct" interpretation of many of the effects which GKS defines to de "device- or implementation-dependent." If such constraint are defined and promulgated, the test suite should be expanded to test for them.

6. The test suites do not test the Computer Graphics Metafile (CGM) and only test the GKS Metafile (GKSM) in a cursory way. Due to the importance of graphical metafiles for the CALS program, work should start at once to develop a test suite for the CGM.

APPENDIX A

SIZE OF LOAD MODULES

Appendix A - Size of Load Modules

This Appendix contains a listing of the size of load modules for individual test programs in the VAX environment. The values are in bytes.

1. Data structure tests

D2011 - (209K)	D2012 - (421K)	D2013 - (486K)
D2014 - (446K)	D2016 - (499K)	D2021 - (541K)
D2022 - (421K)	D2023 - (409K)	D2024 - (457K)
D2026 - (509K)	D2031 - (420K)	D2032 - (435K)
D2033 - (427K)	D2034 - (440K)	D2036 - (497K)
D2041 - (218K)	D2042 - (428K)	D2044 - (442K)
D2051 - (546K)	D2052 - (422K)	D2054 - (437K)
D2056 - (497K)	D2062 - (410K)	D2064 - (437K)
D2072 - (394K)	D2074 - (446K)	D2082 - (395K)
D2084 - (439K)	D2094 - (435K)*	

2. Error Handling Tests

ER0A01 - (370K)	ER0A02 - (459K)	ER0A03 - (397K)
ER0A04 - (383K)	ER0A05 - (372K)	ER0A06 - (422K)
ER0A07 - (379K)	ER0A08 - (372K)	ER0A09 - (371K)
ER0A10 - (371K)	ER0A11 - (370K)	ER0A12 - (400K)
ER0B01 - (427K)	ER0B02 - (380K)	ER0B03 - (407K)
ER0B04 - (403K)	ER0B05 - (394K)	ER1A01 - (397K)
ER1A02 - (394K)	ER1A03 - (468K)	ER1A04 - (464K)
ER1A05 - (404K)	ER1A06 - (468K)	ER1A07 - (465K)
ER1A08 - (367K)	ER1B01 - (401K)	ER1B02 - (392K)
ER2A01 - (461K)*	GKUTIL - (361K)*	

3. Operator Interface Tests

OP0A01 - (444K)	OP0A02 - (407K)	OP0A03 - (459K)
OP0A04 - (464K)	OP0A05 - (460K)	OP0A06 - (445K)
OP0A07 - (466K)	OP0A08 - (453K)	OP0A09 - (458K)
OP0A10 - (494K)	OP0A11 - (468K)	OP0B01 - (462K)
OP0B02 - (467K)	OP0B03 - (458K)	OP0B04 - (453K)
OP0B05 - (454K)	OP1A01 - (468K)	OP1A02 - (406K)
OP1A03 - (441K)	OP1A04 - (492K)	OP1A05 - (488K)
OP1A06 - (480K)	OP1A07 - (490K)	OP1B01 - (509K)
OP2A01 - (481K)*		

(* not linked successfully because we had only a level 1 GKS implementation which did not have the necessary level 2 subroutines)

APPENDIX B

POLYLINE REQUIREMENTS

Appendix B - Requirements Traceability

This appendix contains an analysis of the degree that certain representative GKS requirements are tested by the European validation tests. Each requirement is given a descriptive title and is reference to the ANSI GKS specification by page number. (Paragraphs of the GKS standard contain too much text to be useful for requirements traceability.)

1. **Polyline Bundle Tables** Page: 19, lines -11, -10 (-11 indicates count from bottom of page).

Requirement:
The values in these tables may be dynamically changed. In fact, the only way of changing the aspects of a primitive which are stored in a bundle table is by changing that table.

Test:
Create a polyline primitive with all ASFs set to BUNDLED, and polyline index set to 1. Modify the bundle, using set polyline representation, to markedly different values. With suitable prompts to the operator, check to see that the appearance of the polyline primitive has changed accordingly.

Where Tested:
An equivalent test is performed in Frame 3 of OP1A01.

2. **Polyline Linetypes:** Page: 20, lines 9-15

Requirement:
Linetypes 1 to 4 are solid, dashed, dotted, and dashed dotted. Every workstation of category OUTPUT or OUTIN realizes linetypes 1 to 4 with recognizable styles.

Test:
Create four polylines, one each of linetypes 1 to 4. Documentation or screen text to indicate the linetypes being displayed in their respective locations. Operator checks to see it correct linetypes appear.

Where Tested:
Tested in Frame 1 of OP0A04 of the operator interface tests.

3. **Attribute Binding:** Page : 16, line 28-30

Requirement:
During creation of an output primitive, the attribute values are bound to the primitive and cannot be changed afterwards.

Test:
Somewhat difficult to fully test because of the variety of situations where an incorrect implementation may violate this requirement. A simple test would be to create a polyline with all ASFs set to individual, then change the attribute settings and create another polyline. Check to see that creation of the second primitive did not alter the appearance of the first.

Where Tested:
Equivalent test performed in Frame 1 of OP0A04.

4. **Linewidth:** Page : 22, lines 1-2

Requirement:
The linewidth is calculated as a nominal linewidth multiplied by the linewidth scale factor. This value is mapped by the workstation to the nearest available linewidth.

Test:
The requirement is difficult to fully test, since the set of available linewidths are not obtainable via the GKS inquiry functions. An approximate test would be the output several polylines of various linewidths, with indications to enable the operator to check for at least an approximate proportional relationship between linewidth values and physical width of displayed lines.

Where Tested:
Frame 3 of OP0A04 is a partial test.

5. **Clipping:** Page 51, section 4.7.4.

Requirements:
1) If the clipping indicator in the GKS state list is set to NOCLIP, then primitives put into a segment will have clipping rectangle (0,1) x (0,1) in NDC associated with them.

2) Clipping rectangles are not transformed by the segment transformation, and thus, clipping is always performed against a rectangle whose edges are parallel to the NDC space coordinate axes.

Test:

1) With clipping indicator set to NOCLIP, construct a unit square with POLYLINE in WC with normalization 0 active, and default setting of workstation transformation. The entire unit square should be visible on the display.
2) Set normalization and workstation transformations as in test 1. Set clipping rectangle to (0, 1/2) x (0, 1/2) in NDC. Create a segment containing a polyline drawing of a grid superimposed on the unit square in WC. Close the segment and rotate segment using SET SEGMENT TRANSFORMATION. Verify that the clipping rectangle is unchanged.

Where Tested:

1) Tested in OP0A04, Title Frame.
2) An equivalent test is performed in Frame 2 of OP1A06.

6. Set Polyline Index: Page 89, lines 6-14, section 5.4.1.

Requirements:

1) Sets the "current polyline index" entry in the GKS state list to the value specified by the input parameter.
2) This value to be used when creating subsequent POLYLINE output primitives.
3) Return error error 8 if GKS is in the state GKCL.
4) Return error 60 if the parameter does not belong to the set of integers from 1 to the maximum polyline index for the implementation, inclusive.

Test:

1) Use GQPLR to obtain the list of valid indices. Set the index to a valid value using GSPLI. Use GQPLI to check that this index has been set.
2) Use GQPLR to ascertain the contents of the currently-set polyline index. Output a polyline with all attributes set to "BUNDLED". Check to see if its appearance matches the attributes contained in the currently set index.
3) Set GKS to the state GKCL. Call GSPLI and verify that error 8 has been returned.
4) Use GQEPLI to obtain the list of valid indices. Call GSPLI with with an invalid index. Check to see that error 60 has been returned. For reasonable coverage, a set of invalid

indexes can be used for successive calls to GSPLI. In particular, the values 0, maximum index plus one, and a random invalid parameter should be checked.

Where Tested:

1) Program D2012 of data structure tests.
2) Equivalent test performed in Frame 3 of OP1A01.
3) Tested in Test 1 of ESPLI of ER0A03 of the error tests
4) Only the invalid values 0, -1 are tested in Tests 5 and 6 of ER0A03.

7. Set Linetype: Page 89, lines, 15-32, section 5.4.1.

Requirement:

If the specified linetype is not available on a workstation, linetype 1 is used on that workstation

Test:

Use GQPLF to determine the set of available linetypes. Use GSL to set the linetype to a non-available type. Then output a polyline with this linetype, check to see if it is shown as linetype 1.

Where tested:

Not Tested.

Remark: Frame 2 of OP0A04 makes the comment: "Note that linetype number K should appear the same on all workstations in an implementation". This contradicts the requirement.

8. Set Aspect Source Flags: Page 99, lines 6-27, section 5.4.1.

Requirement:

Set the ASFs for polyline attributes to INDIVIDUAL or BUNDLED.

Test:

Set the polyline index to a specific valid value. Set the individual polyline attributes to values different from those in the specified bundle. Create 2 (or more) polylines, varying the settings of the ASFs. Check to see that the appearance of the polylines varies according to the ASFs.

Where tested:

Tested in OP1A01, Frame 2.

9. Set Polyline Representation: Page 100, section, 5.4.2

Requirement:
Redefine a predefined entry in a bundle table.

Test:
Set polyline index to a predefined bundle. Set all polyline ASFs to "BUNDLED". Create a polyline. Use GSPLR to change the values in the bundle for the current polyline index. Check to see that the appearance of the polyline has changed accordingly.

Where Tested:
An equivalent test is performed in Frame 3 of OP1A01.

10. Set Segment Transformation: Page 115, section 5.6.1.

Requirements:
When a segment is displayed, the coordinates of its primitives are transformed by the transformation matrix given by the parameter in the call to GSSGT.

Test:
Create a segment consisting of a polyline. Call GSSGT with a matrix which rotates and translates the display of the segment.

Where Tested:
Tested in Frame 1 of OP1A06.

APPENDIX C

LISTINGS FROM INSTALLATION

Documentation for Subroutine UWKVAL

3.2.2.2 Details of the sub-program 'UNKVAL'

This sub-program provides the test programs with information about the connection to the workstations in the GKS implementation.

The parameters are as follows:

```
SUBROUTINE UWKVAL (WKID, WKCON, WKTYP)

INTEGER WKID, WKCON, WKTYP
```

This sub-program provides an indicator (WKID) in a table with pairs of connection identifiers and workstation types, which are supported by the GKS implementation Connection identifiers and workstation types, which require this indicator are entered in the parameters WKCON and WKTYP.

```
      ..
!     THIS SECTION MUST BE MODIFIED BY THE TESTER !
!     ------------------------------------------  !
!                                                 !
!  * CHANGE THE PARAMETER VALUE, NUMWK, IF IT IS  !
!    DIFFERENT FROM THE REFERENCE IMPLEMENTATION  !
!    .. PARAMETERS ..                             !
      PARAMETER (NUMWK=2)
!     .. LOCAL ARRAYS ..                          !
      INTEGER LWKCON(NUMWK),LWKTYP(NUMWK)
!     ..                                          !
!  * THE LIST OF CORRECT WORKSTATION TYPES AVAILABLE !
!    ON THE REFERENCE IMPLEMENTATION              !
      DATA LWKTYP/9014,4115/
!     E.G.        4107,SIG,CC,PL1,PL2,HP,HPP,4010,MO,MI !
!                                                 !
!  * THE LIST OF CORRESPONDING WORKSTATION CONNECTION !
!    IDENTIFIERS FOR THE ABOVE WORKSTATION TYPE   !
      DATA LWKCON/10,10/
!                                                 !

      IF (WKID.GT.0 .AND. WKID.LE.NUMWK) THEN
          WKTYP = LWKTYP(WKID)

!     THIS SECTION IS NORMALLY NOT REQUIRE EDITING, !
!     BUT ON SOME SYSTEMS (LIKE THE ICL PERQ), THE  !
!     SPECIFIED WORKSTATION CONNECTION IDENTIFIER   !
!     CAN ONLY BE OBTAINED BY CALLING THE SYSTEM    !
!     UTILITY ROUTINE.                              !
!                                                   !
!     THE FOLLOWING IS AN EXAMPLE TO SHOW HOW THE   !
!     WORKSTATION CONNECTION IDENTIFIER IS OBTAINED !
!     ON THE ICL PERQ.                              !
!                                                   !
          WKCON = LWKCON(WKID)

         CALL UFUNS(3,GKEF,FNAME)
```

```
C     NOTE : THIS IS 7.4 VERSION WITH AN EXTRA PARAMETER
C        CALL GOPKS(GKEF,0)
C        IF (WKID.EQ.1) CALL GCWK('GKSDISP1',WKCON)
C        IF (WKID.EQ.2) CALL GCWK('GKSDISP2',WKCON)
C        IF (WKID.EQ.3) CALL GCWK('GKSDISP1',WKCON)
C        CALL GCLKS
C                                                          :
C     NOTE : THE TWO WINDOW FILES, 'GKSDISP1' AND          :
C     'GKSDISP2' MUST BE EXISTED IN THE CURRENT            :
C     FILE DIRECTORY WHEN IT IS CALLED. DO REMEMBER TO     :
C     REMOVE THE COMMENT CHARACTER FOR THE ABOVE           :
C     EXECUTABLE STATEMENTS.                               :
C-----------------------------------------------------------
      END IF

      RETURN
      END
```

```
      GPE    =
      GVECTR = 0
      GRASTR = 1
      GOTHWK = 2
      GINVIS = 0
      GVISI = 1
      GOUTPT = 0
      GINPUT = 1
      GOUTIN = 2
      GWISS = 3
      GMO = 4
      GMI = 5
      GINACT = 0
      GACTIV = 1
      GPLBND = 0
      GPMBND = 1
      GTXBND = 2
      GFABND = 3
      GLSOLI = 1
      GLDASH = 2
      GLDOT = 3
      GLDASD = 4
      GPOINT = 1
      GPLUS = 2
      GAST = 3
      GOMARK = 4
      GXMARK = 5
*** INITIALISIEREN VON COMONBEREICH PDCTROL    ******************
      STKPT = 0
      TRACE = .FALSE.
*** OPEN DESCRIPTION FILE
      OPEN (LUNDES,FILE='XAIDSR.CIP',ERR=20,IOSTAT=NIOSTA,STATUS='OLD')
      IF (NIOSTA.NE.0) THEN
*** ERROR OR IOSTAT NOT CORRECT
20      WRITE (LUNOUT,'(A)') ' DESCRIPTION FILE COULD NOT BE OPEND '
        WRITE (LUNOUT,'(A)')
     1      ' PLEASE ENTER A LOGICAL UNIT NUMBER TO TRY IT AGAIN (I2)'
        READ (LUNIN,'(I2)') LUNDES
*** SECOND CHANCE OTHERWISE A NEW INITIALISATION
        OPEN (LUNDES,FILE='XAIDSR.CIP',ERR=30,IOSTAT=NIOSTA,
     1          STATUS='OLD')
```

```
        IF (NIOSTA.NE.0) THEN
C *** FILE COULD NOT BE OPEND, NEW INITIALSATION
30        WRITE (LUNOUT,'(A)') ' NEW INTITIALISATION REQUIRED '
          GOTO 10
        ENDIF
      ENDIF
C *** SET LOGICAL UNIT NUMBERS
      READ (LUNDES,'(X,I2,X,I2,X,I2,X,I2)',ERR=10) LUNERR,LUNREP,
     1                                             LUNTRA,LUNGKS
C *** IF ERROR OUCCURS NEW INITIALISATION REQUIRED
      EMPTIA(1) = -32767
      EMPTRA(1) = -9999.99
      RIREL = 2
      CPI = 4

C INITIALISIEREN VON COMONBEREICH CCTROL     ***********************

      MSTKPT = 1
      NBRERR = 0
      EPS = 0.00005
      INDERR = 0

      LOPEN =.FALSE.
C *** OPEN ALL FILES IMPLEMENTATION DEPENDENT
      IF (LUNERR.EQ.LUNTRA) THEN
        IF (LUNERR.EQ.LUNREP) THEN
C *** ONLY ONE FILE REQUESTED
          OPEN (LUNERR,FILE='ERT.VAL',ERR=31,STATUS='UNKNOWN')
          GOTO 32
1         LOPEN =.TRUE.
2         CONTINUE
        ELSE
C *** ERRORS  AND TRACE INTO ONE FILE, REPORT INTO ANOTHER
          OPEN (LUNERR,FILE='ET.VAL',ERR=33,STATUS='UNKNOWN')
          GOTO 34
3         LOPEN =.TRUE.
4         CONTINUE
          OPEN (LUNREP,FILE='R.VAL',ERR=35,STATUS='UNKNOWN')
          GOTO 36
5         LOPEN =.TRUE.
6         CONTINUE
        ENDIF
```

```
      SUBROUTINE ECHEKR(RESULT)

+     GKS CERTIFICATION.........2.0 F77
+     UTILITY SUBROUTINE FOR ERROR TEST SOFTWARE

+ ***************************************************************

      AUTHOR: S.T.YEE   OCT.  84

      PARAMETER :
       LOGICAL -- RESULT (OUTPUT);    TRUE = TEST IS PASSED
                                      FALSE = TEST IS FAILED
+ ***************************************************************

      MAINTENANCE LOG :
      ----------------
      DATE                        DESCRIPTION
      ----                        -----------
      10/12/84                    ADDED AN IF CONDITION TO CHECK
                                  WHEN ONE OR MORE ERRORS ARE
                                  EXPECTED AND NO ERROR REPORTED,
                                  THEN THE TEST IS FAILED

      23/01/85                    RENAME SUBROUTINE CHEKER TO ECHEKR
-----------------------------------------------------------------------
      LOGICAL RESULT

    RESULT:  INDICATES TEST PASS OR FAILURE

      INTEGER TOTAL,NPASS,NFAIL,TYP,NUM,EXPR(10),NER,REP(10),NR,
     :        TNUM
      COMMON/CCERR2/TOTAL,NPASS,NFAIL,TYP,NUM,EXPR,NER,REP,NR,TNUM

    EXPR  : LIST OF EXPECTED ERRORS
    NER   : NUMBER OF EXPECTED ERRORS
    REP   : LIST OF REPORTED ERRORS
    NR    : NUMBER OF REPORTED ERRORS
    NPASS : NUMBER OF TESTS PASSED
    NFAIL : NUMBER OF TESTS FAILED
```

Maintenance log left in delivered test routine

Subroutine ECHEKR

Subroutine ESTART

```
      SUBROUTINE ESTART(TNAME)
C
C +   GKS CERTIFICATION ... ERROR TEST
C +   VERSION OF TEST SUITE ... 2.0 F77
C + *** ************************************************************
C + *** **** IMPLEMENTATION DEPENDENT ROUTINE ********************
C + *** ************************************************************
C +
C + *** ************************************************************
C    ROUTINE TO INITIALIZE COMMONS BLOCKS FOR ALL TEST PROGRAMS
C    AUTHOR: M. GOEBEL & M. MAGUIRE  APR 82
C    MODIFIED BY S.T.YEE OCT. 84
C
C    PARAMETER:
C              (IN)    TNAME CHARACTER*6
C +
C + *** ************************************************************
C
C      MAINTENANCE LOG :
C      ---------------
C      DATE                        DESCRIPTION
C      ----                        -----------
C      23/01/85                    RENAME SUBROUTINE CSTART TO ESTART
C                                  AND ADDED ONE INPUT PARAMETER.
C
C      30/09/85                    HIGHLIGHT THE SECTIONS TO BE
C                                  MODIFIED BY THE TESTER
C +
C + *** ************************************************************
C
C     COMMON BLOCKS FROM EXTERNAL FILES
C
C     INITIALISED IN CSTART.F
C
      COMMON/CC01/RFILE,VDUI,VDUO
      INTEGER RFILE,VDUI,VDUO
C
C
C     VARIABLES ARE INTIALISED IN CSTART.F
C
      COMMON/CC05/GKCL,GKOP,WSOP,WSAC,SGOP
      INTEGER GKCL,GKOP,WSOP,WSAC,SGOP
```

```
C ----------------------------------------------------------------
C :   THIS SECTION MUST BE MODIFIED BY THE TESTER :              :
C :   ---------------------------------------------              :
C :                                                              :
C : CALL PERQ WINDOW MANAGER TO ASSIGN A WORKSTATION             :
C : CONNECTION IDENTIFIER TO THE SPECIFIED WORKSTATION           :
C           CALL GCWK('GKSDISP1',WC1)
C :                                                              :
C ----------------------------------------------------------------
            CSTATE=CSTATE+1
         ENDIF
         IF (CSTATE.EQ.GKOP.AND.CSTATE.LT.RSTATE) THEN
C
C     OPEN SPECIFIED WORKSTATION ..
C
            CALL GOPWK(WI1,WC1,WT1)
            CSTATE=CSTATE+1
         ENDIF
C
C     ACTIVATE SPECIFIED WORKSTATION...
         IF (CSTATE.EQ.WSOP AND.CSTATE.LT.RSTATE)  CALL GACWK(WI1)
C
C     OPERATING STATE 'SGOP' IS EXAMINED WITHIN THE TEST PROGRAMS
      ELSE
C
C     DECREMENT CURRENT OPERATING STATE
         IF (CSTATE.EQ.WSAC) THEN
C
C     DEACTIVATE SPECIFIED WORKSTATION
            CALL GDAWK(WI1)
            CSTATE=CSTATE-1
         ENDIF
         IF (CSTATE.EQ.WSOP.AND.CSTATE.GT.RSTATE) THEN
C
C     CLOSE SPECIFIED WORKSTATION
            CALL GCLWK(WI1)
C
C ----------------------------------------------------------------
C :   THIS SECTION MUST BE MODIFIED BY THE TESTER :              :
C :   ---------------------------------------------              :
C :                                                              :
```

```
C ! DISCONNECT THE WINDOW -- PERQ WORKSTATION ONLY          !
C            CALL GDWK(WC1)
C !                                                          !
C ------------------------------------------------------------
             CSTATE=CSTATE-1
          ENDIF
          IF(CSTATE.EQ.GKOP.AND.CSTATE.GT.RSTATE) THEN
C
C   CHECK IF THERE ARE WORKSTATIONS THAT HAVE NOT BEEN CLOSED BEFORE
C   GKS IS CLOSE DOWN.
C
             CALL GQOPWK(1,IERR,IL,LWKID(1))
             IF(IL.NE.0.AND.IERR.EQ.0) THEN
             DO 1000 I=1,IL
               CALL GQOPWK(I,IERR,OL,LWKID(I))
               CALL GCLWK(LWKID(I))
1000         CONTINUE
             ENDIF
             CALL GCLKS
          ENDIF
       ENDIF
C
 9999  CONTINUE
       RETURN
       END
```

```
      INTEGER NWK
      CHARACTER TEST*6
C     ..
C     .. ARRAY ARGUMENTS ..
      INTEGER IDATA(*)
C     ..
C     .. LOCAL SCALARS ..
      INTEGER I, IN, KFONT, OUT, WKID
      CHARACTER FNAME*80
C
C ---------------------------------------------------------
C |   THIS SECTION MUST BE MODIFIED BY THE TESTER :       |
C |   ---------------------------------------------       |
C |                                                       |
C |   NAME OF THE REFERENCE IMPLEMENTATION                |
      CHARACTER IMPLE*30
      INTEGER MAXWK
C |                                                       |
      DATA IMPLE/'ISSCO GKS : '/,
     *     MAXWK/2/
C |                                                       |
C ---------------------------------------------------------
C
C
C     ..
C     .. EXTERNAL SUBROUTINES ..
C     EXTERNAL UFUNS
C     ..
      CALL UFUNS(1, IN, FNAME)
      CALL UFUNS(2, OUT, FNAME)
C
C     HEADING
C
      WRITE (OUT, 9000) TEST
 9000 FORMAT (1x, 'GKS TEST SUITE ... VERSION 2.0 ... TEST ', A)
      WRITE (OUT, 9010) IMPLE
 9010 FORMAT (1x, 'IMPLEMENTATION : ', A)
C
C     SELECT WORKSTATION FOR TESTING
C
      I = 1
```

```
      INTEGER       IFLAG,WKPT,I,OL,SGNA(100),ISGNA(11)
      CHARACTER*6  NAME
C *** WRITE SUBROUTINE NAME ON PROGRAM STACK
      DATA   NAME/'D2013 '/
C *** INITIALISATION
      CALL PDINIT
      STKPT = STKPT + 1
      STK(STKPT) = NAME
C *** ADD CURRENT MESSAGESTACKPOINTER TO MESSAGEPOINTERSTACK
      MPTSTK (STKPT) = MSTKPT
C *** TRACE START
      IF (TRACE) THEN
      CALL PDTRNP(1)
      ENDIF
C *** TRACE END                                                        ***
C *** ****************    EFFECT    ***********************************
      CALL CWRITE(NAME,'PROGRAM D2013 STARTED',21)
C *** CURRENT TEST LEVEL HIGH ENOUGH?
      IF (TSTLEV.GE GL1A) THEN
C *** TEST LEVEL O.K.
       CALL DINIT
C *** LOOP OVER ALL ACTIVE WORKSTATIONS
       DO 50,WKPT=1,NAVWK
       IF (LWKAC(WKPT).AND.(WKCA(WKPT) NE GINPUT.AND WKCA(WKPT).NE.
     1     GMI)) THEN
C *** TOP 1. : CHECK LIST IS  EMPTY
          CALL CPUT (1,NAME,1411)
          CALL GQSGWK (WKPT,1,INDERR,OL,SGNA)
          IF (INDERR.NE.0.AND.OL.NE.0) THEN
            CALL CERR (1,NAME,INDERR)
          ELSE IF (OL.NE.0) THEN
           CALL CERR(1,NAME,1411)
          ENDIF
         ENDIF
   50  CONTINUE
C *** TOP 2. : CREATE FIRST SEGMENT
         CALL GCRSG (1)
         CALL GCLSG
         ISGNA(1)=1
C
C *** ONE SEGMENT ON EVERY WORKSTATION ??
       DO 150,WKPT=1,NAVWK
```

Program D2013

```
     1      GMI)) THEN
         CALL CQSGWK (WKPT,1,ISGNA,IFLAG)
         ENDIF
 150   CONTINUE
C *** TOP 3.:LOOP OVER TEN DIFFERENT SEGMENTS
         DO 200,I=1,10
          CALL GCRSG (2*I)
          CALL GCLSG
 200     CONTINUE
C *** ADD SEGMENTS NAME INTO LIST OF EXPECTED SEGMENTS
         DO 220,I=1,10
          ISGNA(I+1)=2*I
 220     CONTINUE
C *** ONE SEGMENT ON EVERY WORKSTATION ??
        DO 250,WKPT=1,NAVWK
        IF (LWKAC(WKPT).AND.(WKCA(WKPT).NE.GINPUT.AND.WKCA(WKPT).NE.
     1      GMI)) THEN
         CALL CQSGWK (WKPT,11,ISGNA,IFLAG)
        ENDIF
 250    CONTINUE
        CALL DTERM
       ELSE
        CALL CWRITE (NAME,' TEST COULD NOT EXECUTED, CURRENT TEST LEVEL
     + TOO LOW',52)
       ENDIF
       CALL CWRITE (NAME,' TEST PROGRAM D2013 FINISHED ',29)

C *** ***********************************************************************
C *** TRACE START
       IF (TRACE) THEN
       CALL PDTRNP(2)
       ENDIF
C *** TRACE END
C *** SET CURRENT MESSAGESTACKPOINTER
       MSTKPT = MPTSTK(STKPT)
C *** REMOVE SUBROUTINENAME FROM PROGRAMSTACK
       STKPT = STKPT - 1
       CALL CNUMER
       STOP 'PROGRAMM D2013 FINISHED'
       END
```

```
      CALL UFUNS(4,FUNIT,FNAME)

      OPEN TEST REPORT FILE WHIC HAS THE SAME FILENAME OF THE TEST FILE
      WITH FILE EXTENSION '.TXT'
      FNAME(1:10)=TNAME(1:6)//'.TXT'
      OPEN (UNIT=FUNIT,status='new',FILE=FNAME(1:10))
      RFILE=FUNIT

      WRITE HEADING TO REPORT FILE

      WRITE(RFILE,105)TNAME(1:6),IMPLE
105   FORMAT(///'       GKS TEST SUITE ...  VERSION 2.0 ....  TEST ',A,
     :///'       === IMPLEMENTATION :  ',A,' ==='//)

      GET CHANNEL NUMBERS FOR DIALOG

      CALL UFUNS(1,IN,KEYBRD)
      CALL UFUNS(2,OUT,KEYBRD)
      VDUI=IN
      VDUO=OUT

      OUTPUT HEADING ON VDU ...

      WRITE(OUT,100)TNAME(1:6),IMPLE
100   FORMAT(///'  GKS TEST SUITE ...  VERSION 2.0 ....  TEST ',A,
     :///'       === IMPLEMENTATION :  ',A,' ==='//)

    CCOS INITIALIZATION
    -------------------
      GKCL=0
      GKOP=1
      WSOP=2
      WSAC=3
      SGOP=4

    CCWORK INITIALIZATION
    ---------------------
    ERROR FILE UNIT
      CALL UFUNS(3,GKSEF,KEYBRD)
      GKEF=GKSEF
-------------------------------------------------------------
!     THIS SECTION MUST BE MODIFIED BY THE TESTER :        !
```

APPENDIX D

LISTINGS FROM DATA STRUCTURE TESTS

```
C ***********************************************************************
C LOA TESTPROGRAM                                                      *** 1
C ***                                                                  ***
C *** PARAMETER:   NONE                                                ***
C ***                                                                  ***
C ***                                                                  ***
C *** EFFECT:  THE GKS-STATE-LIST WILL BE TESTED FOR WORKSTATION       ***
C ***          INDEPENDED POLYLINE ATTRIBUTES                          *** 2
C ***                                                                  ***
C *** ERRORS:                                                          ***
C ***          1388 ERROR:  TRY TO SET POLYLINE INDEX                  ***
C ***          1389 ERROR:  TRY TO SET LINETYPE                        ***
C ***          1390 ERROR:  TRY TO SET LINEWIDTH SCALE VECTOR          *** 3
C ***                                                                  ***
C *** PROJECT:  VALGKS.APPLICATION                                     *** 5
C ***                                                                  ***
C *** AUTHOR:  F.G.GESSERT        CREATIONDATE:  JAN/17/83             *** 4
C ***                                                                  ***
C *** UPDATE:  F.G.GESSERT                DATE:  AUG/20/85             ***
C ***          B.F.KIRSCH                                              *** 5
C ***                                                                  ***
C *** VERSION :  - FORTRAN 77                                          ***
C ***            -- SMALLER PACKAGES                                   ***
C ***            -- NO RANDOM VALUES FOR PARAMETERS                    ***
C ***            - FOR GKS 7.4                                         *** 6
C ***                                                                  ***
C ***********************************************************************

C *** DECLARATIONS
      COMMON /PDCTRL/ STK(20), STKPT, TRACE, LUNOUT, LUNIN, LUNERR,    7
     *                LUNTRA, EMPTIA(1), EMPTRA(1)
     *                ,RIREL,CPI,LUNREP,LUNGKS                         7
                                                                       11
C *** PROGRAMSTACK AND PROGRAMSTACKPOINTER
      INTEGER         STKPT
      CHARACTER*6     STK
C *** FLAG FOR TRACEMODE                                               4
      LOGICAL         TRACE
C *** LOGICAL UNITNUMBERS FOR OUTPUT, INPUT, ERROR, TRACE              4
      INTEGER         LUNOUT, LUNIN, LUNERR, LUNTRA
C *** DUMMIES FOR  NOT USED PARAMETERS                                 8
      INTEGER         EMPTIA
```

```
      REAL                EMPTRA
C *** REAL INTEGER RELATION
      INTEGER             RIREL
C *** CHARACTER PER INTEGER
      INTEGER             CPI
C *** LOGICAL UNIT NUMBER REPORT AND GKS
      INTEGER             LUNREP,LUNGKS
      COMMON /CCTROL/ EPS,I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,IA1,IA2,IA3,   ⇦ 10
     +                INDERR,ISEVER,MPTSTK,MSTKPT,NBRERR,
     +   ⇦ 11         LEVEL,MSGERR,
     +                R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,RA1,RA2
C *** INTERN INTEGER VARIABLES                                          ⇦ 9
      INTEGER             I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,IA1(100),IA2(50),
     +                IA3(300)
C *** ERROPINDICATOR,MESSAGEPOINTERSTACK,MESSAGESTACKPOINTER,
C *** INTERN SYSTEMTRACELEVEL,NUMBER OF ERRORS,
C *** ERRORMESSAGECOLLECTION
      INTEGER             INDERR,ISEVER,MPTSTK(10),MSTKPT,
     +                LEVEL,MSGERR(150),NBRERR
C *** DISTANCELIMIT FOR COMPARISON OF REALS,INTERN REALVARIABLES
      REAL                EPS,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,RA1(50),RA2(50)
      COMMON /CCMNEM/   GBUNDL,GINDIV,GCONDI,GALWAY,GNCLIP,GCLIP,       ⇦ 10
     +                GMONOC,GCOLOR,GWC,GNDC,GASAP,GBNIL,GBNIG,GASTI,
     +                GUNDET,GDETEC,GMETRE,GOTHU,GNEMPT,GEMPTY,
     +                GIRG,GIMM,GNECHO,GECHO,
     +                GHOLLO,GSOLID,GPATTR,GHATCH,GNORML,GHILIT,
     +                GNONE,GOK,GNPICK,GNCHOI,
     +                GNCLAS,GLOCAT,GSTROK,GVALUA,GCHOIC,GPICK,GSTRIN,
     +                GSUPPD,GALLOW,
     +                GLOA,GLOB,GLOC,GL1A,GL1B,GL1C,GL2A,GL2B,GL2C,
     +                GNO,GYES,GREQU,GSAMPL,GEVENT,
     +                GGKCL,GGKOP,GWSOP,GWSAC,GSGOP,GABSNT,GPRSNT,
     +                GSUPP,GPERFO,GHIGHR,GLOWER,GNMORE,GMORE,
     +                GAHNOR,GALEFT,GACENT,GARITE,
     +                GAVNOR,GATOP,GACAP,GAHALF,GABASE,GABOTT,
     +                GRIGHT,GLEFT,GUP,GDOWN,GSTRP,GCHARP,GSTRKP,
     +                GSET,GREALI,GNPEND,GPEND,GVECTR,GRASTR,GOTHWK,
     +                GINVIS,GVISI,GOUTPT,GINPUT,GOUTIN,GWISS,GMO,GMI,
     +                GINACT,GACTIV,GPLBND,GPMBND,GTXBND,GFABND,
     +                GLSOLI,GLDASH,GLDOT,GLDASD,
     +                GPOINT,GPLUS,GAST,GOMARK,GXMARK
      INTEGER           GBUNDL,GINDIV,GCONDI,GALWAY,GNCLIP,GCLIP,
```

```
     +                    GMONOC, GCOLOR, GWC, GNDC, GASAP, GBNIL, GBNIG, GASTI,
     +                    GUNDET, GDETEC, GMETRE, GOTHU, GNEMPT, GEMPTY,
     +                    GIRG, GIMM, GNECHO, GECHO,
     +                    GHOLLO, GSOLID, GPATTR, GHATCH, GNORML, GHILIT,
     +                    GNONE, GOK, GNPICK, GNCHOI,
     +                    GNCLAS, GLOCAT, GSTROK, GVALUA, GCHOIC, GPICK, GSTRIN,
     +                    GSUPPD, GALLOW,
     +                    GLOA, GLOB, GLOC, GL1A, GL1B, GL1C, GL2A, GL2B, GL2C,
     +                    GNO, GYES, GREQU, GSAMPL, GEVENT,
     +                    GGKCL, GGKOP, GWSOP, GWSAC, GSGOP, GABSNT, GPRSNT,
     +                    GSUPP, GPERFO, GHIGHR, GLOWER, GNMORE, GMORE,
     +                    GAHNOR, GALEFT, GACENT, GARITE,
     +                    GAVNOR, GATOP, GACAP, GAHALF, GABASE, GABOTT,
     +                    GRIGHT, GLEFT, GUP, GDOWN, GSTRP, GCHARP, GSTRKP,
     +                    GSET, GREALI, GNPEND, GPEND, GVECTR, GRASTR, GOTHWK,
     +                    GINVIS, GVISI, GOUTPT, GINPUT, GOUTIN, GWISS, GMO, GMI,
     +                    GINACT, GACTIV, GPLBND, GPMBND, GTXBND, GFABND,
     +                    GLSOLI, GLDASH, GLDOT, GLDASD,
     +                    GPOINT, GPLUS, GAST, GOMARK, GXMARK
C
C *** NUMBER OF AVAILABLE WORKSTATION TYPES.
C *** PARAMETER FOR NUMBER OF AVAILABLE WORKSTATIONS ⇦ 12
      INTEGER NAVWK
      PARAMETER (NAVWK=46)
C *** COMMON FOR WORKSTATION TYPES, CONNECTIONS AND CATEGORIES
      COMMON /CCWKTY/ WKTP,WKCO,WKCA,LWKAC,TSTLEV,IBUF
C *** LIST OF WORKSTATION TYPES, LIST OF CONNECTION IDENTIFIERS
      INTEGER WKTP(NAVWK),WKCO(NAVWK),WKCA(NAVWK),TSTLEV,IBUF
      LOGICAL LWKAC(NAVWK)
      COMMON /CCCTRL/ MSGSTK,WKNAME
C
CC *** COMMON CONTAINING ALL CHARACTER VARIABLES
C *** MESSAGESTACK
      CHARACTER*80     MSGSTK(10)
C *** NAMES AF SUPPORTED WORKSTATIONS
      CHARACTER*20     WKNAME(46)
C *** 46 IS THE SIZE OF NAVWK (PARAMETER IN CCWKTY)
C *** INTERN VARIABLES
      INTEGER          IFLAG,WKFT,I,NLT,LT,NLW,NPPLI
      REAL             NOMLW,RLWMIN,RLWMAX
      CHARACTER*6      NAME
C *** WRITE SUBROUTINE NAME ON PROGRAM STACK
```

```
      DATA   NAME/'D2012 '/
C *** INITIALISATION
      CALL PDINIT
      STKPT = STKPT + 1
      STK(STKPT) = NAME
C *** ADD CURRENT MESSAGESTACKPOINTER TO MESSAGEPOINTERSTACK
      MPTSTK (STKPT) = MSTKPT
C *** TRACE START
      IF (TRACE) THEN
      CALL PDTRNP(1)                                  13
      ENDIF
C *** TRACE END
C *** ******************   EFFECT   ***********************************
      CALL CWRITE(NAME, 'PROGRAM D2012 STARTED', 21)
      CALL DINIT
C *** LOOP OVER ALL ACTIVE WORKSTATIONS                12
      DO 100, WKPT=1, NAVWK
      IF (LWKAC(WKPT). AND. (WKCA(WKPT). EQ. GOUTPT. OR. WKCA(WKPT). EQ.
     1     GOUTIN)) THEN
C *** INQUIRE RANGE OF PARAMETER VALUES                13
      CALL GQPLF(WKTP(WKPT), 1, INDERR, NLT, LT, NLW, NOMLW, RLWMIN, RLWMAX,
     1           NPPLI)
C *** INQUIRY FAILED, PROGRAM TERMINATION
      IF (INDERR. NE. 0) THEN
       CALL CWRITE (NAME, ' INQUIRY -GQPLF- FAILED ', 24)
       CALL CWRITE (NAME, ' TEST PROGRAM FAILED ', 21)
       CALL CERR(1, NAME, INDERR)
       GOTO 999
      ENDIF
C *** SET POLYLINE INDEX
      CALL CPUT (1, NAME, 1388)
C *** SET MINIMAL INDEX                               14
      CALL DSPLI(1, IFLAG)                            16
C *** SET MAXIMUM INDEX
      CALL DSPLI(NPPLI, IFLAG)                        15
C *** SET VALUE IN BETWEEN MAX-MIN
      CALL DSPLI((1+NPPLI)/2, IFLAG)
C *** SET LINETYPE
      CALL CPUT(1, NAME, 1389)
C *** SET LINETYPE DASH-DOTTED
      CALL DSLN(CLDASD, IFLAG)
C *** SET LINETYPE LOWER ZERO
```

```
      IF (NLT.GT.4) CALL DSLNC(-NLT+4,IFLAG)
C *** SET LINEWIDTH SCALE FACTOR
      CALL CPUT (1,NAME,1390)
C *** SET MINIMAL LINEWIDTH SCALEFACTOR
      CALL DSLWSC(RLWMIN/NOMLW,IFLAG)
C *** SET MAXIMAL LINEWIDTH SCALEFACTOR
      CALL DSLWSC(RLWMAX/NOMLW,IFLAG)
C *** SET LINEWIDTH SCALE FACTOR 1.5
      CALL DSLWSC (1.5,IFLAG)
      ENDIF
100   CONTINUE
      CALL DTERM
      CALL CWRITE (NAME,' TEST PROGRAM D2012 FINISHED ',29)
C
C *** ***********************************************************************
  999 CONTINUE
C *** TRACE START
      IF (TRACE) THEN
      CALL PDTRNP(2)
      ENDIF
C *** TRACE END
C *** SET CURRENT MESSAGESTACKPOINTER
      MSTKPT = MPTSTK(STKPT)
C *** REMOVE SUBROUTINENAME FROM PROGRAMSTACK
      STKPT = STKPT - 1
      CALL CNUMER
      STOP 'PROGRAMM D2012 FINISHED'
      END
```

17

```
      SUBROUTINE DSPLCI (ICOLI,IDFLAG)                                  07773
      COMMON /PDCTRL/ STK(50), STKPT, TRACE, LUNOUT, LUNIN, LUNERR,
     +                LUNTRA, EMPTIA(1), EMPTRA(1)
     +                ,RIREL,CPI,LUNREP,LUNGKS
C *** PROGRAMSTACK AND PROGRAMSTACKPOINTER
      INTEGER         STKPT
      CHARACTER*6     STK
C *** FLAG FOR TRACEMODE
      LOGICAL         TRACE
C *** LOGICAL UNITNUMBERS FOR OUTPUT, INPUT, ERROR, TRACE
      INTEGER         LUNOUT, LUNIN, LUNERR, LUNTRA
C *** DUMMIES FOR  NOT USED PARAMETERS
      INTEGER         EMPTIA
      REAL            EMPTRA
C *** REAL INTEGER RELATION
      INTEGER         RIREL
C *** CHARACTER PER INTEGER
      INTEGER         CPI
C *** LOGICAL UNIT NUMBER REPORT AND GKS
      INTEGER         LUNREP,LUNGKS
      COMMON /CCTROL/ EPS,I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,IA1,IA2,IA3,
     +                INDERR,ISEVER,MPTSTK,MSTKPT,NBRERR,
     +                LEVEL,MSGERR,
     +                R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,RA1,RA2
C *** INTERN INTEGER VARIABLES
      INTEGER         I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,IA1(100),IA2(50),
     +                IA3(300)
C *** ERRORINDICATOR,MESSAGEPOINTERSTACK,MESSAGESTACKPOINTER,
C *** INTERN SYSTEMTRACELEVEL,NUMBER OF ERRORS,
C *** ERRORMESSAGECOLLECTION
      INTEGER         INDERR,ISEVER,MPTSTK(10),MSTKPT,
     +                LEVEL,MSGERR(150),NBRERR
C *** DISTANCELIMIT FOR COMPARISON OF REALS,INTERN REALVARIABLES
      REAL            EPS,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,RA1(50),RA2(50)
      COMMON /CCCTRL/ MSGSTK,WKNAME
C
CC *** COMMON CONTAINING ALL CHARACTER VARIABLES
C *** MESSAGESTACK
      CHARACTER*80    MSGSTK(10)
C *** NAMES AF SUPPORTED WORKSTATIONS
      CHARACTER*20    WKNAME(46)
C *** 46 IS THE SIZE OF NAWNK (PARAMETER IN CCWKTY)
```

```
      INTEGER        ILTYPE,MSERR,ICOLPL,IMTYPE,ICOLPM,IFONT,IPREC
      INTEGER        ICOLIT,INTS,ISTYLI,ICOLIF,IFLAG
      REAL           RLWIDT,RMSZSF,RCHXP,RCHSP
      CHARACTER*6    NAME
      DATA NAME /'DSPLCI'/
      STKPT = STKPT + 1
      STK(STKPT) = NAME
      MPTSTK (STKPT) = MSTKPT
      IF( NOT (TRACE)) GO TO 99999
      CALL PDTR4I(1,ICOL1,EMPTIA(1),EMPTIA(1),EMPTIA(1))
99999 IDFLAG = 0
      CALL GQLWSC(MSERR,RLWIDT)
      IF(.NOT.(MSERR.NE.0)) GO TO 99998
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99998 CALL GQPLCI(MSERR,ICOLPL)
      IF(.NOT.(MSERR.NE.0)) GO TO 99997
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99997 CALL GQMK(MSERR,IMTYPE)
      IF(.NOT.(MSERR.NE.0)) GO TO 99996
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99996 CALL GQMKSC(MSERR,RMSZSF)
      IF(.NOT.(MSERR.NE.0)) GO TO 99995
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99995 CALL GQPMCI(MSERR,ICOLPM)
      IF(.NOT.(MSERR.NE.0)) GO TO 99994
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99994 CALL GQTXFP(MSERR,IFONT,IPREC)
      IF(.NOT.(MSERR.NE.0)) GO TO 99993
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99993 CALL GQCHXP(MSERR,RCHXP)
      IF(.NOT.(MSERR.NE.0)) GO TO 99992
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99992 CALL GQCHSP(MSERR,RCHSP)
      IF(.NOT.(MSERR.NE.0)) GO TO 99991
```

```
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99991 CALL GQTXCI(MSERR,ICOLIT)
      IF(.NOT.(MSERR.NE.0)) GO TO 99990
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99990 CALL GQFAIS(MSERR,INTS)
      IF(.NOT.(MSERR.NE.0)) GO TO 99989
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99989 CALL GQFASI(MSERR,ISTYLI)
      IF(.NOT.(MSERR.NE.0)) GO TO 99988
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99988 CALL GQFACI(MSERR,ICOLIF)
      IF(.NOT.(MSERR.NE.0)) GO TO 99987
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99987 CALL GQLN(MSERR,ILTYPE)
      IF(.NOT.(MSERR.NE.0)) GO TO 99986
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99986 CALL GSPLCI(ICOLI)
      CALL CPUT(1,NAME,1323)
      CALL CQPLCI(ICOLI,IFLAG)
      IF(.NOT.(IFLAG.EQ.1)) GO TO 99985
      IDFLAG=4
      CALL CWRITE(NAME,'FUNCTION DSPLCI FAILED',23)
99985 CALL CPUT (1,NAME,1299)
      CALL CQLWSC(RLWIDT,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99984
      IDFLAG=2
      CALL CERR (1,NAME,1273)
99984 CALL CQMK(IMTYPE,IFLAG)
      IF( NOT (IFLAG.NE.0)) GO TO 99983
      IDFLAG=2
      CALL CERR(1,NAME,1276)
99983 CALL CQMKSC(RMSZSF,IFLAG)
      IF( NOT.(IFLAG.NE.0)) GO TO 99982
      IDFLAG=2
      CALL CERR(1,NAME,1277)
99982 CALL CQPMCI(ICOLPM,IFLAG)
```

```
      IF(.NOT.(IFLAG.NE.0)) GO TO 99981
      IDFLAG=2
      CALL CERR(1,NAME,1278)
99981 CALL CQTXFP(IFONT,IPREC,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99980
      IDFLAG=2
      CALL CERR(1,NAME,1280)
99980 CALL CQCHXP(RCHXP,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99979
      IDFLAG=2
      CALL CERR(1,NAME,1281)
99979 CALL CQCHSP(RCHSP,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99978
      IDFLAG=2
      CALL CERR(1,NAME,1282)
99978 CALL CQTXCI(ICOLIT,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99977
      IDFLAG=2
      CALL CERR(1,NAME,1283)
99977 CALL CQFAIS(INTS,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99976
      IDFLAG=2
      CALL CERR(1,NAME,1287)
99976 CALL CQFASI(ISTYLI,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99975
      IDFLAG=2
      CALL CERR(1,NAME,1288)
99975 CALL CQFACI(ICOLIF,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99974
      IDFLAG=2
      CALL CERR(1,NAME,1289)
99974 CALL CQLN(ILTYPE,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99973
      IDFLAG=2
      CALL CERR(1,NAME,1272)
99973 IF(.NOT.(TRACE)) GO TO 99972
      CALL PDTR4I(2,EMPTIA(1),EMPTIA(1),EMPTIA(1),EMPTIA(1))
99972 MSTKPT = MPTSTK(STKPT)
      STKPT = STKPT - 1
      RETURN
      END
```

```
      SUBROUTINE DSPLI (IINDEX,IDFLAG)                                  07967
      COMMON /PDCTRL/ STK(20), STKPT, TRACE, LUNOUT, LUNIN, LUNERR,
     *                LUNTRA, EMPTIA(1), EMPTRA(1)
     *                ,RIREL,CPI,LUNREP,LUNGKS
C *** PROGRAMSTACK AND PROGRAMSTACKPOINTER
      INTEGER         STKPT
      CHARACTER*6     STK
C *** FLAG FOR TRACEMODE
      LOGICAL         TRACE
C *** LOGICAL UNITNUMBERS FOR OUTPUT, INPUT, ERROR, TRACE
      INTEGER         LUNOUT, LUNIN, LUNERR, LUNTRA
C *** DUMMIES FOR  NOT USED PARAMETERS
      INTEGER         EMPTIA
      REAL            EMPTRA
C *** REAL INTEGER RELATION
      INTEGER         RIREL
C *** CHARACTER PER INTEGER
      INTEGER         CPI
C *** LOGICAL UNIT NUMBER REPORT AND GKS
      INTEGER         LUNREP,LUNGKS
      COMMON /CCTROL/ EPS,I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,IA1,IA2,IA3,
     +                INDERR,ISEVER,MPTSTK,MSTKPT,NBRERR,
     +                LEVEL,MSGERR,
     +                R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,RA1,RA2
C *** INTERN INTEGER VARIABLES
      INTEGER         I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,IA1(100),IA2(50),
     +                IA3(300)
C *** ERRORINDICATOR,MESSAGEPOINTERSTACK,MESSAGESTACKPOINTER,
C *** INTERN SYSTEMTRACELEVEL,NUMBER OF ERRORS,
C *** ERRORMESSAGECOLLECTION
      INTEGER         INDERR,ISEVER,MPTSTK(10),MSTKPT,
     +                LEVEL,MSGERR(150),NBRERR
C *** DISTANCELIMIT FOR COMPARISON OF REALS,INTERN REALVARIABLES
      REAL            EPS,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,RA1(50),RA2(50)
      COMMON /CCCTRL/ MSGSTK,WKNAME
C
CC *** COMMON CONTAINING ALL CHARACTER VARIABLES
C *** MESSAGESTACK
      CHARACTER*80    MSGSTK(10)
C *** NAMES AF SUPPORTED WORKSTATIONS
      CHARACTER*20    WKNAME(46)
C *** 46 IS THE SIZE OF NAVWK (PARAMETER IN CCWKTY)
```

```
      INTEGER        IINDEX, IDFLAG
      INTEGER        ILTYPE,MSERR,ICOLPL,IMTYPE,ICOLPM,IFONT,IPREC
      INTEGER        ICOLIT,INTS,ISTYLI,ICOLIF,IFLAG
      REAL           RLWIDT,RMSZSF,RCHXP,RCHSP,SLWIDT
      CHARACTER*6    NAME
      DATA NAME /'DSPLI '/
      STKPT = STKPT + 1
      STK(STKPT) = NAME
      MPTSTK (STKPT) = MSTKPT
      IF(.NOT.(TRACE)) GO TO 99999
      CALL PDTR4I(1,ILTYPE,EMPTIA(1),EMPTIA(1),EMPTIA(1))
99999 IDFLAG = 0
      CALL GQLWSC(MSERR,RLWIDT)
      IF(.NOT.(MSERR.NE.0)) GO TO 99998
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99998 CALL GQPLCI(MSERR,ICOLPL)
      IF(.NOT.(MSERR.NE.0)) GO TO 99997
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99997 CALL GQMK(MSERR,IMTYPE)
      IF(.NOT.(MSERR.NE.0)) GO TO 99996
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99996 CALL GQMKSC(MSERR,RMSZSF)
      IF(.NOT.(MSERR.NE.0)) GO TO 99995
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99995 CALL GQPMCI(MSERR,ICOLPM)
      IF(.NOT.(MSERR.NE.0)) GO TO 99994
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99994 CALL GQTXFP(MSERR,IFONT,IPREC)
      IF(.NOT.(MSERR.NE.0)) GO TO 99993
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99993 CALL GQCHXP(MSERR,RCHXP)
      IF(.NOT.(MSERR.NE.0)) GO TO 99992
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99992 CALL GQCHSP(MSERR,RCHSP)
      IF(.NOT.(MSERR.NE.0)) GO TO 99991
```

```
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99991 CALL GQTXCI(MSERR,ICOLIT)
      IF(.NOT.(MSERR.NE.0)) GO TO 99990
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99990 CALL GQFAIS(MSERR,INTS)
      IF(.NOT.(MSERR.NE.0)) GO TO 99989
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99989 CALL GQFASI(MSERR,ISTYLI)
      IF(.NOT.(MSERR.NE.0)) GO TO 99988
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99988 CALL GQFACI(MSERR,ICOLIF)
      IF(.NOT.(MSERR.NE.0)) GO TO 99987
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99987 CALL GQLN(MSERR,ILTYPE)
      IF(.NOT.(MSERR.NE.0)) GO TO 99986
      CALL CERR(1,NAME,MSERR)
      IDFLAG=1
99986 CALL QSPLI(IINDEX)
      CALL CPUT(1,NAME,1347)
      CALL CQPLI(IINDEX,IFLAG)
      IF(.NOT.(IFLAG.EQ.1)) GO TO 99985
      IDFLAG=4
      CALL CWRITE(NAME,'FUNCTION DSPLI FAILED',22)
99985 CALL CPUT (1,NAME,1295)
      CALL CQLWSC(RLWIDT,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99984
      IDFLAG=2
      CALL CERR (1,NAME,1273)
99984 CALL CQPLCI(ICOLPL,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99983
      IDFLAG=2
      CALL CERR(1,NAME,1274)
99983 CALL CQMK(IMTYPE,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99982
      IDFLAG=2
      CALL CERR(1,NAME,1276)
99982 CALL CQMKSC(RMSZSF,IFLAG)
```

```
      IF(.NOT.(IFLAG.NE.0)) GO TO 99981
      IDFLAG=2
      CALL CERR(1,NAME,1277)
99981 CALL CQPMCI(ICOLPM,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99980
      IDFLAG=2
      CALL CERR(1,NAME,1278)
99980 CALL CQTXFP(IFONT,IPREC,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99979
      IDFLAG=2
      CALL CERR(1,NAME,1280)
99979 CALL CQCHXP(RCHXP,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99978
      IDFLAG=2
      CALL CERR(1,NAME,1281)
99978 CALL CQCHSP(RCHSP,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99977
      IDFLAG=2
      CALL CERR(1,NAME,1282)
99977 CALL CQTXCI(ICOLIT,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99976
      IDFLAG=2
      CALL CERR(1,NAME,1283)
99976 CALL CQFAIS(INTS,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99975
      IDFLAG=2
      CALL CERR(1,NAME,1287)
99975 CALL CQFASI(ISTYLI,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99974
      IDFLAG=2
      CALL CERR(1,NAME,1288)
99974 CALL CQFACI(ICOLIF,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99973
      IDFLAG=2
      CALL CERR(1,NAME,1289)
99973 CALL CQLN(ILTYPE,IFLAG)
      IF(.NOT.(IFLAG.NE.0)) GO TO 99972
      IDFLAG=2
      CALL CERR(1,NAME,1272)
99972 IF(.NOT.(TRACE)) GO TO 99971
      CALL PDTR4I(2,EMPTIA(1),EMPTIA(1),EMPTIA(1),EMPTIA(1))
99971 MSTKPT = MPTSTK(STKPT)
```

```
STKPT = STKPT - 1                                    0B163
RETURN                                               0B164
END                                                  0B165
```

```
C *** INDEX EQUAL MAXIMAL INDEX
        CALL DSCR (WKPT,MCOLI-1,.1,.1,.1,IFLAG)
       ENDIF
300    CONTINUE
       CALL DTERM
      ELSE
       CALL CWRITE (NAME,' TEST COULD NOT EXECUTED, CURRENT TEST LEVEL
     + TOO LOW',52)
      ENDIF
      CALL CWRITE (NAME,' TEST PROGRAM D2024 FINISHED ',29)
C
C *** ******************************************************************
C *** TRACE START
      IF (TRACE) THEN
      CALL PDTRNP(2)
      ENDIF
C *** TRACE END
C *** SET CURRENT MESSAGESTACKPOINTER
      MSTKPT = MPTSTK(STKPT)
C *** REMOVE SUBROUTINENAME FROM PROGRAMSTACK
      STKPT = STKPT - 1
      CALL CNUMER
      STOP 'PROGRAMM D2024 FINISHED'
      END
```

```
+++++++++++++++++  1.  ERROR DETECTED +++++++++++++

   1 D2024  : 1024 EXPECTEC FILL AREA REP. BUNDLE NOT DELIVERED
   2 DSFAR  : 1223 DATA ERROR : SET FILL AREA REPRESENTATION
     GQFAR  : 0081 CURRENT GKS ERROR NUMBER
 + DSFAR  :    DSFAR FAILED
+++++++++++++++++  2.  ERROR DETECTED +++++++++++++

   1 D2024  : 1025 EXPECTED PATTERN BUNDLE REP. NOT DELIVERED
   2 DSPAR  : 1224 DATA ERROR : SET PATTERN REPRESENTATION
     GQPAR  : 0090 CURRENT GKS ERROR NUMBER
 + DSPAR  :    DSPAR FAILED
+++++++++++++++++  3.  ERROR DETECTED +++++++++++++

   1 D2024  : 1025 EXPECTED PATTERN BUNDLE REP. NOT DELIVERED
   2 DSPAR  : 1224 DATA ERROR : SET PATTERN REPRESENTATION
     GQPAR  : 0085 CURRENT GKS ERROR NUMBER
 + DSPAR  :    DSPAR FAILED
+++++++++++++++++  4.  ERROR DETECTED +++++++++++++

   1 D2024  : 1025 EXPECTED PATTERN BUNDLE REP. NOT DELIVERED
   2 DSPAR  : 1224 DATA ERROR : SET PATTERN REPRESENTATION
     GQPAR  : 0085 CURRENT GKS ERROR NUMBER
 + DSPAR  :    DSPAR FAILED
+++++++++++++++++  5.  ERROR DETECTED +++++++++++++

   1 D2024  : 1026 EXPECTEC COLOUR REPRESENTATION NOT DELIVERED
   2 DSCR   : 1225 DATA ERROR : SET COLOUR REPRESENTATION
     GQCR : 0094 CURRENT GKS ERROR NUMBER
 + DSCR   :    DSCR FAILED
+++++++++++++++++  6.  ERROR DETECTED +++++++++++++

   1 D2024  : 1026 EXPECTEC COLOUR REPRESENTATION NOT DELIVERED
   2 DSCR   : 1225 DATA ERROR : SET COLOUR REPRESENTATION
     GQCR : 0094 CURRENT GKS ERROR NUMBER
 + DSCR   :    DSCR FAILED
 + D2024  :     TEST PROGRAM D2024 FINISHED

-------------------------------------------------
NUMBER OF ERRORS DETECTED :      6
-------------------------------------------------
```

Program D2024
Error report file

```
      STKPT = STKPT + 1
      MPTSTK (STKPT) = MSTKPT
      STK(STKPT) = NAME
C *** TRACE START
      IF (TRACE) THEN
         CALL PDTRNP(1)
      ENDIF
C *** TRACE END                                                        ***
C *** ******************    EFFECT    ***********************************
C
C
      CALL CWRITE(NAME,'PROGRAM D2011 -DOPKS  STARTED',30)
      CALL CPUT (1,NAME,1350)
      CALL DOPKS (LUNGKS,IBUF,IFLAG)
      IF (IFLAG.NE.0) THEN
        CALL CWRITE (NAME,' DOPKS FAILED  -- GECLKS CALLED ',33)
        CALL CWRITE(NAME,'GKS EMERGENCY CLOSED',20)
        CALL GECLKS
      ELSE
        CALL CPUT(1,NAME,1351)
        CALL DCLKS (IIFLAG)
        IF (IIFLAG.EQ.0) THEN
          CALL CWRITE(NAME,'TEST PROGRAM D2011 PERFORMED',28)
        ELSE
          CALL CWRITE(NAME,' DCLKS FAILED --  GECLKS CALLED ',33)
          CALL CWRITE(NAME,'GKS EMERGENCY CLOSED',20)
          CALL GECLKS
        ENDIF
      ENDIF
C *** ***********************************************************************
C
C *** TRACE START
      IF (TRACE) THEN
        CALL PDTRNP(2)
      ENDIF
C *** TRACE END
C *** SET CURRENT MESSAGESTACKPOINTER
      MSTKPT = MPTSTK(STKPT)
C *** REMOVE SUBROUTINENAME FROM PROGRAMSTACK
      STKPT = STKPT - 1
      CALL CNUMER
      STOP 'TESTPROGRAM D2011 FINISHED'
```

Program D2011

```
      RAWX (1) = 0.0                                                    02743
      RAWX (2) = 1.0                                                    02744
      RAWX (3) = 0.0                                                    02745
      RAWX (4) = 1.0                                                    02746
      DO 99964 J=1,IMAXTR                                               02747
      CALL CPUT (1,NAME,1154)                                           02748
      CALL CQNT (J-1,RAWX,RAWX,IFLAG)                                   02749
      IF (IFLAG.NE.0) IDFLAG = 2                                        02750
99964 CONTINUE                                                          02751
99968 IF(.NOT.(TSTLEV.GE.GL1A)) GO TO 99963                             02755
      CALL GQACWK (1,INDERR,I1,I2)                                      02757
      IF(.NOT.(INDERR.NE.0.AND.I1.NE.0)) GO TO 99961                    02758
      CALL CERR(1,'GQACWK',INDERR)                                      02758
      GO TO 99962                                                       02758
99961 IF (I1.NE.0) CALL CERR(1,NAME,1156)                               02760
99962 CONTINUE                                                          02762
99963 CONTINUE                                                          02766
 990  CONTINUE                                                          02766
      IF (TRACE) CALL PDTR4I (2,IDFLAG,EMPTIA(1),EMPTIA(1),EMPTIA(1))   02768
      MSTKPT=MPTSTK(STKPT)                                              02771
      STKPT = STKPT - 1                                                 02773
      RETURN                                                            02774
      END                                                               02775
```

```
C
      CALL DOPKS (LUNGKS,IBUF,IFLAG)
C
      IF (IFLAG.NE.0) THEN
        CALL CWRITE (NAME,' DOPKS AFTER CLOSE GKS FAILED    ',33)
        CALL CWRITE(NAME,'GKS EMERGENCY CLOSED',20)
        CALL GECLKS
      ELSE
        CALL GCLKS
      ENDIF
C
        IF (IIFLAG.EQ.0) THEN
          CALL CWRITE(NAME,'TEST PROGRAM D2041 PERFORMED',28)
        ENDIF
C
C *** ***********************************************************************
C
C *** TRACE START
      IF (TRACE) THEN
        CALL PDTRNP(2)
      ENDIF
C *** TRACE END
C *** SET CURRENT MESSAGESTACKPOINTER
      MSTKPT = MPTSTK(STKPT)
C *** REMOVE SUBROUTINENAME FROM PROGRAMSTACK
      STKPT = STKPT - 1
      CALL CNUMER
      STOP 'TESTPROGRAM D2041 FINISHED'
      END
```

Program D2041

Program D2011 &
Program D2041
Error Report File

```
+++++++++++++++++  1.  ERROR DETECTED +++++++++++++

   1 D2041   : 1350 PROGRAM START WITH ERRORS
   2 DOPKS   : 1150 DEFAULT PATTERN SIZE NOT DELIVERED
     CGPA    : 1092 EXPECTED PATTERN WIDTH Y-DIR NOT DELIVERED
               EXPECTED:   0.00000E+00
               DELIVERED:    1.0000
+++++++++++++++++++++++++++++++++++++++++++++

+++++++++++++++++  2.  ERROR DETECTED +++++++++++++

   1 D2041   : 1350 PROGRAM START WITH ERRORS
   2 DOPKS   : 1150 DEFAULT PATTERN SIZE NOT DELIVERED
     CGPA    : 1093 EXPECTED PATTERN HEIGHT Y-DIR NOT DELIVERED
               EXPECTED:    1.0000
               DELIVERED:   0.00000E+00
+++++++++++++++++++++++++++++++++++++++++++++

+++++++++++++++++  3.  ERROR DETECTED +++++++++++++

   1 D2041   : 1350 PROGRAM START WITH ERRORS
   2 DOPKS   : 1152 DEFAULTPICK IDENTIFIER NOT DELIVERED
     CGPKID  : 1000 EXPEXTED VALUE NOT DELIVERED
               EXPECTED:              0
               DELIVERED:              1
+++++++++++++++++++++++++++++++++++++++++++++

+++++++++++++++++  4.  ERROR DETECTED +++++++++++++

   1 D2041   : 1350 PROGRAM START WITH ERRORS
     GGACWK  : 2002 CURRENT GKS LANGUAGE BINDING ERROR NUMBER
 + D2041   :     DOPKS AFTER CLOSE GKS FAILED
 + D2041   :    GKS EMERGENCY CLOSED
 + D2041   :    TEST PROGRAM D2041 PERFORMED

---------------------------------------------
NUMBER OF ERRORS DETECTED :     4
---------------------------------------------

---------------------------------------------
SUPPORTED TEST LEVEL      : 1B
---------------------------------------------
```

```
      INTEGER       WKAC(NAVWK),WKOP(NAVWK)
      CHARACTER*6   NAME
      DATA    NAME/'D2021 '/
C *** INITIALISATION
      CALL PDINIT
C *** WRITE SUBROUTINENAME ONTO PROGRAMSTACK
      STKPT = STKPT + 1
      MPTSTK (STKPT) = MSTKPT
      STK(STKPT) = NAME
C *** TRACE START
      IF (TRACE) THEN
         CALL PDTRNP(1)
      ENDIF
C *** TRACE END                                                        ***
C *** ******************   EFFECT   ************************************
C
C INITIALISATION AND DIALOG
      CALL CWRITE(NAME,'PROGRAM D2021 -DOPWK, DACWK STARTED',36)
      CALL CPUT (1,NAME,1350)
      CALL GOPKS (LUNGKS,IBUF)
C
C
      NWKOP = 0
      NWKAC = 0
      COUNT = 0
      NWK = 0
      IDFLAG = 0
C
C
C --    THE ARRAYS WKOP AND WKAC HAVE DIMENSION NAVWK
C       THIS IS SET IN COMMON CCWKTY
C
C*** OPEN ALL WORKSTATIONS  *************
C
      DO  500, LOOPWK =1,NAVWK
      CALL CPUT(2,NAME,1424)
      IF (LWKAC(LOOPWK)) THEN
         NWK = NWK +1
         WKID = LOOPWK
         CALL DOPWK(WKID,WKCO(WKID),WKTP(WKID),IDFLAG)
C
         IF (IDFLAG.LT.3) THEN
```

Program D2021

```
      CALL CPUT (1,NAME,1023)
C *** INDEX EQUAL FIRST INDEX
      CALL DSTXR (WKPT,1,1,GSTRP,2.,1.,1,IFLAG)
C *** INDEX IN BETWEEN MIN MAX
      CALL DSTXR (WKPT,(MTXBTE+1)/2,1,GCHARP,.2,.1,MCOLI/2,IFLAG)
C *** INDEX EQUAL MAXIMAL INDEX
      CALL DSTXR (WKPT,MTXBTE,1,GSTRKP,.5,-.1,MCOLI-1,IFLAG)
C *** TOP 4.: SET FILL AREA REPRESENTATION
      CALL CPUT (1,NAME,1024)
C *** INDEX EQUAL FIRST INDEX
      CALL DSFAR (WKPT,1,GHOLLO,1,1,IFLAG)
C *** INDEX IN BETWEEN MIN MAX
      CALL DSFAR (WKPT,(MFABTE+1)/2,GSOLID,1,1,IFLAG)
C *** INDEX EQUAL MAXIMAL INDEX
      CALL DSFAR (WKPT,MFABTE,GPATTR,MPAI,MCOLI-1,IFLAG)
C *** TOP 5.: SET PATTERN REPRESENTATION
      CALL CPUT (1,NAME,1025)
C *** DEFINE A COLOUR INDEX ARRAY
      COUNT=0
      DO 200,I=1,DIMX
       DO 100,J=1,DIMY
        IF (COUNT.LT.MCOLI-1) THEN
         COUNT=COUNT+1
        ELSE
         COUNT=0
        ENDIF
        COLIA (I,J)=COUNT
100    CONTINUE
200   CONTINUE
C *** INDEX EQUAL FIRST INDEX
      CALL DSPAR (WKPT,1,DIMX,DIMY,1,1,2,2,COLIA,IFLAG)
C *** INDEX IN BETWEEN MIN MAX
      CALL DSPAR (WKPT,(MPAI+1)/2,DIMX,DIMY,DIMX/2,DIMY/2,DIMX/4,
     1            DIMY/4,COLIA,IFLAG)
C *** INDEX EQUAL MAXIMAL INDEX
      CALL DSPAR (WKPT,MPAI,DIMX,DIMY,1,1,DIMX,DIMY,COLIA,IFLAG)
C *** TOP 6.: SET COLOUR REPRESENTATION
      CALL CPUT (1,NAME,1026)
C *** INDEX EQUAL FIRST INDEX
      CALL DSCR (WKPT,1,1.,1.,1.,IFLAG)
C *** INDEX IN BETWEEN MIN MAX
      CALL DSCR (WKPT,MCOLI/2,.5,.5,.5,IFLAG)
```

Program D2024

```
      CALL CERR (1,'GQDLC ',INDERR)
      IF (IDFLAG.EQ.0) IDFLAG = 1
      GO TO 99925
99924 IF(.NOT.(I7.LT.1)) GO TO 99923
      CALL CERR (1,NAME,1252)
      IF (IDFLAG.LT.1) IDFLAG = 1
99923 CALL CPUT (1,NAME,1253)
      CALL CQLCS (IWKID,I,GSET,100,GREQU,GECHO,0,R1,R2,1,REAREA,I9,
     +            KDAREC,IFLAG)
      IF (IDFLAG.LT.IFLAG) IDFLAG = 2
99925 CONTINUE
99926 CONTINUE
99927 IF(.NOT.(I2.NE.0)) GO TO 99922
      DO 99921 I=1,I2
      CALL GQSKS (IWKID,I,GSET,25,100,INDERR,GREQU,GECHO,ITNR,INP,
     +            RA1,RA2,1,REAREA,I7,I8,KDAREC)
      CALL GQDSK (IWTYPE,I,1,100,INDERR,I7,I8,I9,REAREA,I10,I1,KDAREC)
      IF(.NOT.(INDERR.NE.0)) GO TO 99919
      CALL CERR (1,'GQDSK ',INDERR)
      IF (IDFLAG.EQ.0) IDFLAG = 1
      GO TO 99920
99919 IF(.NOT.(I8.LT.1)) GO TO 99918
      CALL CERR (1,NAME,1252)
      IF (IDFLAG.LT.1) IDFLAG = 1
99918 CALL CPUT (1,NAME,1254)
      CALL CQSKS (IWKID,I,GSET,25,100,GREQU,GECHO,0,INP,RA1,
     +            RA2,1,REAREA,I10,I1,KDAREC,IFLAG)
      IF (IDFLAG.LT.IFLAG) IDFLAG = 2
99920 CONTINUE
99921 CONTINUE
99922 IF(.NOT.(I3.NE.0)) GO TO 99917
      DO 99916 I=1,I3
      CALL GQDVL (IWTYPE,I,1,100,INDERR,R1,I7,I8,REAREA,R2,R3,I9,KDAREC)
      IF(.NOT.(INDERR.NE.0)) GO TO 99914
      CALL CERR (1,'GQDVL ',INDERR)
      IF (IDFLAG.EQ.0) IDFLAG = 1
      GO TO 99915
99914 IF(.NOT.(I7.LT.1)) GO TO 99913
      CALL CERR (1,NAME,1252)
      IF (IDFLAG.LT.1) IDFLAG = 1
99913 CALL CPUT (1,NAME,1255)
      CALL CQVLS (IWKID,I,100,GREQU,GECHO,R1,1,REAREA,R2,R3,I9
```

```
      REAL          RCR,RCG,RCB                                       04885
      INTEGER       IFLAG                                             04887
      CHARACTER*6    NAME                                             04888
      DATA NAME /'DSCR   '/                                           04890
      STKPT = STKPT + 1                                               04891
      STK(STKPT) = NAME                                               04892
      MPTSTK (STKPT) = MSTKPT                                         04894
      IF(.NOT.(TRACE)) GO TO 99999                                    04896
      CALL PDTR4I(1,IWKID,ICI,EMPTIA(1),EMPTIA(1))                    04897
      CALL PDTR4R(3,RCR,RCG,RCB,EMPTRA(1))                            04898
99999 IDFLAG=0                                                        04902
      CALL GSCR(IWKID,ICI,RCR,RCG,RCB)                                04903
      CALL CPUT (1,NAME,1225)                                         04904
      CALL CGCR(IWKID,ICI,GSET,RCR,RCG,RCB,IFLAG)                     04905
      IF(.NOT.(IFLAG.EQ.1)) GO TO 99998                               04906
      IDFLAG=4                                                        04907
      CALL CWRITE(NAME,'DSCR FAILED',11)                              04908
99998 IF(.NOT.(TRACE)) GO TO 99997                                    04912
      CALL PDTR4I(2,IDFLAG,EMPTIA(1),EMPTIA(1),EMPTIA(1))             04913
99997 MSTKPT = MPTSTK(STKPT)                                          04917
      STKPT = STKPT - 1                                               04919
      RETURN                                                          04920
      END                                                             04921
```

Subroutine DSCR

```
+++++++++++++++++  1.  ERROR DETECTED +++++++++++++

   1 D2024  :  1024 EXPECTEC FILL AREA REP.  BUNDLE NOT DELIVERED
   2 DSFAR  :  1223 DATA ERROR : SET FILL AREA REPRESENTATION
     GQFAR  :  0081 CURRENT GKS ERROR NUMBER
 + DSFAR  :    DSFAR FAILED
+++++++++++++++++  2.  ERROR DETECTED +++++++++++++

   1 D2024  :  1025 EXPECTED PATTERN BUNDLE REP.  NOT DELIVERED
   2 DSPAR  :  1224 DATA ERROR : SET PATTERN REPRESENTATION
     GQPAR  :  0090 CURRENT GKS ERROR NUMBER
 + DSPAR  :    DSPAR FAILED
+++++++++++++++++  3.  ERROR DETECTED +++++++++++++

   1 D2024  :  1025 EXPECTED PATTERN BUNDLE REP.  NOT DELIVERED
   2 DSPAR  :  1224 DATA ERROR : SET PATTERN REPRESENTATION
     GQPAR  :  0085 CURRENT GKS ERROR NUMBER
 + DSPAR  :    DSPAR FAILED
+++++++++++++++++  4.  ERROR DETECTED +++++++++++++

   1 D2024  :  1025 EXPECTED PATTERN BUNDLE REP.  NOT DELIVERED
   2 DSPAR  :  1224 DATA ERROR : SET PATTERN REPRESENTATION
     GQPAR  :  0085 CURRENT GKS ERROR NUMBER
 + DSPAR  :    DSPAR FAILED
+++++++++++++++++  5.  ERROR DETECTED +++++++++++++

   1 D2024  :  1026 EXPECTEC COLOUR REPRESENTATION NOT DELIVERED
   2 DSCR   :  1225 DATA ERROR : SET COLOUR REPRESENTATION
     GQCR : 0094 CURRENT GKS ERROR NUMBER
 + DSCR   :    DSCR FAILED
+++++++++++++++++  6.  ERROR DETECTED +++++++++++++

   1 D2024  :  1026 EXPECTEC COLOUR REPRESENTATION NOT DELIVERED
   2 DSCR   :  1225 DATA ERROR : SET COLOUR REPRESENTATION
     GQCR : 0094 CURRENT GKS ERROR NUMBER
 + DSCR   :    DSCR FAILED
 + D2024  :     TEST PROGRAM D2024 FINISHED

-------------------------------------------------
NUMBER OF ERRORS DETECTED :      6
-------------------------------------------------
```

Program D2024
Error Report File

APPENDIX E

LISTINGS FROM ERROR TESTS

```
W2=.TRUE.

 INITIALISE INVALID WORKSTATION CATEGORIES
   WOUT=-1
   WIN=-1
   WOI=-1
   WISS=-1
   WMO=-1
   WMI=-1

SET WORKSTATION CATEGORY

   IF(WKCAT1.EQ.0) WOUT=WI1
   IF(WKCAT1.EQ.1) WIN=WI1
   IF(WKCAT1.EQ.2) WOI=WI1
   IF(WKCAT1.EQ.3) WISS=WI1
   IF(WKCAT1.EQ.4) WMO=WI1
   IF(WKCAT1.EQ.5) WMI=WI1

 CCERR2 INITIALIZATION
 ---------------------
   TOTAL=0
   NPASS=0
   NFAIL=0

 INVALID TEXT INITIALIZATION
 ---------------------------
   E101=.FALSE.
   LINV=3
   CHINV='    '

   WRITE(VDUO,'(/A)') ' '
   WRITE(VDUO,'(/A)') 'TEST IS RUNNING ............'

   RETURN
   END
```

```
C
      INTEGER I,J,NP,NF
      LOGICAL PASS,FAIL
      DATA PASS,FAIL/.TRUE.,.FALSE./
C
      NP=NPASS
      NF=NFAIL
C
      RESULT=PASS
C
C  CHECK IF NO ERROR IS EXPECTED AND REPORTED
C
      IF(NR.EQ.0.AND.NER.EQ.0) THEN
         NPASS=NPASS+1
      ELSE IF(NR.GT.NER) THEN
C
C  TEST FAILED IF NUMBER OF REPORTED ERRORS ARE GREATER THAN EXPECTED
         RESULT=FAIL
         NFAIL=NFAIL+1
      ENDIF
C
C  CHECK NP/NF HAS BEEN INCREASE BY 1
      IF (NFAIL.GT.NF.OR.NPASS.GT.NP) GOTO 9990
C
C  CHECK STATE ERROR IS EXPECTED
C
      type *,expr(1),rep(1)
      IF(EXPR(1).LT.10) THEN
        IF(EXPR(1).NE.REP(1)) THEN
C
C  TEST FAILED IF STATE ERROR IS NOT REPORTED
           RESULT=FAIL
           NFAIL=NFAIL+1
        ELSE IF(NER.EQ.1.AND.NR.EQ.1) THEN
C
C  STATE ERROR IS REPORTED AND THERE IS ONLY ONE ERROR EXPECTED
C
           NPASS=NPASS+1
        ENDIF
      ENDIF
C
C  CHECK NP/NF HAS BEEN INCREASED BY 1
```

Subroutine ECHEKR

```
      IF FAI T.I OR. SS. NP OTI 990

CHECK ERRORS REPORTED MUST BE EXPECTED ERRORS

      J=0

CHECK IF NO ERROR REPORTED, TEST FAIL...

      IF(NR.EQ.J) GOTO 400
00    J=J+1
      I=1
      type *,expr(i),rep(j)
      IF (J.GT.NR) GOTO 9000
00    IF (REP(J).EQ.EXPR(I)) GOTO 200
      I=I+1
      IF (I.LE.NER) GOTO 300

TEST FAILED AS REPORTED ERRORS ARE NOT EXPECTED

00    RESULT=FAIL
      NFAIL=NFAIL+1

      GOTO 9990

ALL REPORTED ERRORS ARE EXPECTED

000   NPASS=NPASS+1
990   RETURN

      END
```

Subroutine ECHEKR

```
      PROGRAM EROA02
C-----------------------------------------------------
C   CHECK ERROR GENERATION FOR OUTPUT FUNCTIONS
C   AUTHOR: M. MAGUIRE  APR 82;     S.T. YEE    OCT 84
C   NO: EROA02
C-----------------------------------------------------
C
C     COMMON BLOCKS FROM EXTERNAL FILES
C
C     INITIALISED IN CSTART.F
C
      COMMON/CC01/RFILE,VDUI,VDUO
      INTEGER RFILE,VDUI,VDUO
C
C
      CALL ESTART('EROA02')
C
C
C   WRITE TITLE AND INITIALIZE TEST RUN
C
      WRITE(RFILE,100)
C 100 FORMAT(1X,56(1H*)
C     *//1X,58HTEST PROGRAM EROA02 - ERROR REPORTING FOR OUTPUT FUNCTIONS
C     *//1X,58(1H*))
  100 FORMAT(1X,58('*')
     *//1X,'TEST PROGRAM EROA02 - ERROR REPORTING FOR OUTPUT FUNCTIONS'
     *//1X,58('*'))
C
C   POLYLINE TEST SET: OA-201
      CALL EPL
C
C   POLYMARKER TEST SET: OA-202
      CALL EPM
C
C   TEXT TEST SET: OA-203
      CALL ETX
C
C   FILL AREA TEST SET: OA-204
      CALL EFA
C
C   CELL ARRAY TEST SET: OA-205
      CALL ECA
```

Program EROA02

```
      CALL GCA(PX,PY,QX,QY,3,3,3,3,COL)
      CALL ECHEKR(RESULT)
      IF (RESULT) NUM=NUM+1
      IF(.NOT.RESULT) CALL EREP(RESULT)

TYPE: PARAMETER ERROR CHECK
      TYP=2

TEST NO: 5
      TNUM=5
      CALL ESET(WSAC)
      EE(1)=91
      CALL ESEXER(1,EE)
      CALL GCA(PX,PY,QX,QY,0,0,1,1,0,0,COL)
      CALL ECHEKR(RESULT)
      IF (RESULT) NUM=NUM+1
      IF(.NOT.RESULT) CALL EREP(RESULT)


TYPE: MULTIPLE ERROR CHECK
      TYP=3

TEST NO: 6
      TNUM=6
      CALL ESET(WSOP)
      EE(1)=5
      EE(2)=91
      CALL ESEXER(2,EE)
      CALL GCA(PX,PY,QX,QY,0,0,1,1,0,0,COL)
      CALL ECHEKR(RESULT)
      IF (RESULT) NUM=NUM+1
      IF(.NOT.RESULT) CALL EREP(RESULT)

REPORT TEST PASSED/FAILED
      TOTAL=TOTAL+6
      RESULT=.TRUE.
      CALL EREP(RESULT)

      RETURN
      END
```

```
  CALL ESEXER(1,EE)

  CURRENT FORTRAN BINDING
  CALL GQPXA(WI1,PX,PY,NMX,MMX,1,1,N,M,EIND,INVVAL,COL)

  PREVIOUS BINDING ..
  CALL GQPXA(WI1,PX,PY,NMX,MMX,M,EIND,INVVAL,COL)  ⇦ 1
  NR=0
  IF(EIND.NE.0)NR=1
  REP(1)=EIND
  CALL ECHEKR(RESULT)
  IF (RESULT) NUM=NUM+1
  IF(.NOT.RESULT) CALL EREP(RESULT)

TEST NO: 2
  TNUM=2
  CALL ESET(GKOP)
  EE(1)=7
  CALL ESEXER(1,EE)

  CURRENT FORTRAN BINDING
  CALL GQPXA(WI1,PX,PY,NMX,MMX,1,1,N,M,EIND,INVVAL,COL)

  PREVIOUS BINDING ..
   CALL GQPXA(WI1,PX,PY,NMX,MMX,M,EIND,INVVAL,COL)  ⇦ 1
  NR=0
  IF(EIND.NE.0)NR=1
  REP(1)=EIND
  CALL ECHEKR(RESULT)
  IF (RESULT) NUM=NUM+1
  IF(.NOT.RESULT) CALL EREP(RESULT)

TEST NO: 3
  TNUM=3
  CALL ESET(WSOP)
  IF(WIN.NE.-1.OR.WISS.NE.-1.OR.WMO.NE.-1.OR.WMI.NE.-1) THEN
    EE(1)=39
    CALL ESEXER(1,EE)
  ELSE IF(WTPXR.EQ.-1) THEN
    EE(1)=40
    CALL ESEXER(1,EE)
  ELSE
```

This program is shown after modification to call the correct (current) FORTRAN binding, rather than the incorrect (previous) one called in the source code furnished us.

```
C-----------------------------------------------------------
C   CHECK ERROR GENERATION FOR INQUIRY FUNCTIONS
C   AUTHOR: A. SILVA  APR 82;   S.T.YEE   OCT 84
C   NO: EROA12
C-----------------------------------------------------------
C
C
C     INITIALISED IN CSTART.F
C
      COMMON/CC01/RFILE,VDUI,VDUO
      INTEGER RFILE,VDUI,VDUO
C
C
      CALL ESTART('EROA12')
C
      WRITE(RFILE,100)
  100 FORMAT(1X,66('*')
     */1X,'*  TEST PROGRAM EROA12 ',
     *     '- ERROR REPORTING OF INQUIRY FUNCTIONS,    *'
     */1X,'*',23X,'PIXEL INQUIRIES',26X,'*'
     */1X,66('*'))
C
C
C   INQUIRY FUNCTIONS FOR PIXELS
C   ----------------------------
C
C   INQUIRE PIXEL ARRAY DIMENSIONS TEST SET: OA-1201
      CALL EQPXAD
C
C   INQUIRE PIXEL ARRAY TEST SET: OA-1202
      CALL EQPXA
C
C   INQUIRE PIXEL TEST SET: OA-1203
      CALL EQPX
C
C   CLOSE GKS...
C
      CALL ESET(0)
C
      CALL ESUM
C
```

Program EROA12

```
        CALL ESEXER(0,EE)
      ENDIF
C
C     CURRENT FORTRAN BINDING
      CALL GQPXA(WI1,PX,PY,NMX,MMX,1,1,N,M,EIND,INVVAL,COL)
C
C     PREVIOUS BINDING ..
C      CALL GQPXA(WI1,PX,PY,NMX,MMX,M,EIND,INVVAL,COL)  1
      NR=0
      IF(EIND.NE.0)NR=1
      REP(1)=EIND
      CALL ECHEKR(RESULT)
      IF (RESULT) NUM=NUM+1
      IF(.NOT.RESULT) CALL EREP(RESULT)
C
C   TEST NO:  4
      TNUM=4
      CALL ESET(WSAC)
      IF(WIN.NE.-1.OR.WISS.NE.-1.OR.WMO.NE.-1.OR.WMI.NE.-1) THEN
        EE(1)=39
        CALL ESEXER(1,EE)
      ELSE IF(WTPXR.EQ.-1) THEN
        EE(1)=40
        CALL ESEXER(1,EE)
      ELSE
        CALL ESEXER(0,EE)
      ENDIF
C
C     CURRENT FORTRAN BINDING
      CALL GQPXA(WI1,PX,PY,NMX,MMX,1,1,N,M,EIND,INVVAL,COL)
C
C     PREVIOUS BINDING ..
C     CALL GQPXA(WI1,PX,PY,NMX,MMX,M,EIND,INVVAL,COL)  1
      NR=0
      IF(EIND.NE.0)NR=1
      REP(1)=EIND
      CALL ECHEKR(RESULT)
      IF (RESULT) NUM=NUM+1
      IF(.NOT.RESULT) CALL EREP(RESULT)
C
C   TYPE: PARAMETER ERROR CHECK
      TYP=2
```

```
C
C    TEST NO:  5
       TNUM=5
       CALL ESET(WSOP)
       EE(1)=20
       CALL ESEXER(1,EE)
C
C      CURRENT FORTRAN BINDING
       CALL GQPXA(INVWID,PX,PY,NMX,MMX,1,1,N,M,EIND,INVVAL,COL)
C
C      PREVIOUS BINDING ..
c       CALL GQPXA(INVWID,PX,PY,NMX,MMX,M,EIND,INVVAL,COL)  1
       NR=0
       IF(EIND.NE.0)NR=1
       REP(1)=EIND
       CALL ECHEKR(RESULT)
       IF (RESULT) NUM=NUM+1
       IF(.NOT.RESULT) CALL EREP(RESULT)
C
C    TEST NO:  6
       TNUM=6
       CALL ESET(WSOP)
C
C    IF SPECIFIED WORKSTATION TYPE HAS NO PIXEL READBACK FACILITY, WTPXR
C    IS SET TO -1 THEN ERROR 40 WOULD BE EXPECTED
C    ERROR 25 IS RETURN IF SPECIFIED WORKSTATION IS NOT OPEN

       IF(WTPXR.EQ.-1) THEN
         EE(1)=40
         EE(2)=25
         CALL ESEXER(2,EE)
       ELSE
         EE(1)=25
         CALL ESEXER(1,EE)
       ENDIF
C
C      CURRENT FORTRAN BINDING
       CALL GQPXA(WI2,PX,PY,NMX,MMX,1,1,N,M,EIND,INVVAL,COL)
C
C      PREVIOUS BINDING ..
c      CALL GQPXA(WI2,PX,PY,NMX,MMX,M,EIND,INVVAL,COL)  1
```

```
      NR=0
      IF(EIND.NE.0)NR=1
      REP(1)=EIND
      CALL ECHEKR(RESULT)
      IF (RESULT) NUM=NUM+1
      IF(.NOT.RESULT) CALL EREP(RESULT)

C     TEST NO: 7
      TNUM=7
      CALL ESET(WSOP)
      IF(WTPXR.EQ.-1) THEN
        EE(1)=40
        EE(2)=84
        CALL ESEXER(2,EE)
      ELSE
        EE(1)=84
        CALL ESEXER(1,EE)
      ENDIF

      CURRENT FORTRAN BINDING
      CALL GQPXA(WI1,PX,PY,0,-1,1,1,N,M,EIND,INVVAL,COL)   <- 2

      PREVIOUS BINDING ..
      CALL GQPXA(WI1,PX,PY,0,-1,M,EIND,INVVAL,COL)   <- 1
      NR=0
      IF(EIND.NE.0)NR=1
      REP(1)=EIND
      CALL ECHEKR(RESULT)
      IF (RESULT) NUM=NUM+1
      IF(.NOT.RESULT) CALL EREP(RESULT)

C     REPORT TEST PASSED/FAILED
      TOTAL=TOTAL+7
      RESULT=.TRUE.
      CALL EREP(RESULT)

      RETURN
      END
      SUBROUTINE EQPX
C-----------------------------------------------------------------
C     CHECK INQUIRE PIXEL ERROR REPORTING
C     AUTHOR: A. SILVA  MAY 83,   S.T.YEE     OCT 84
```

```
      PROGRAM EROB01
C-----------------------------------------------------------------
C   CHECK ERROR GENERATION FOR INITIALISE INPUT DEVICES FUNCTIONS
C   AUTHOR:   S.T.YEE   FEB 85
C   NO: EROB01
C-----------------------------------------------------------------
C
C
C     INITIALISED IN CSTART.F
C
      COMMON/CC01/RFILE,VDUI,VDUO
      INTEGER RFILE,VDUI,VDUO
C
C
C     VARIABLES ARE INTIALISED IN CSTART.F
C
      COMMON/CC05/GKCL,GKOP,WSOP,WSAC,SGOP
      INTEGER GKCL,GKOP,WSOP,WSAC,SGOP
C
C
      CALL ESTART('EROB01')
C
      WRITE(RFILE,100)
  100 FORMAT(1X,80('*')/1X,'*',78X,'*',
     */1X,'* TEST PROGRAM EROB01 ',
     *  ' - ERROR REPORTING OF INITIALISE INPUT DEVICES FUNCTIONS *',
     */1X,'*',78X,'*'/1X,80('*'))
C
C   INITIALISE INPUT DEVICES FUNCTIONS
C   ----------------------------------
C
C   INITIALISE LOCATOR TEST SET: OB-101
      CALL EINLC
C
C   INITIALISE CHOICE TEST SET: OB-102
      CALL EINCH
C
C   INITIALISE STRING TEST SET: OB-103
      CALL EINST
C
C   INITIALISE STROKE TEST SET: OB-104
      CALL EINSK
```

Program EROB01

```
  YMAX=SRY

INQUIRE DEFAULT STROKE DEVICE DATA
  CALL GQDSK(WT1,SKDNR,1,MLDR,ERR,MBUF,NPET,NTHPET,EAREA,BUFLN,LDR,
 :     DATREC)

INITIALISE PARAMETERS
  IPX(1)=0.5
  IPY(1)=0.5
  ITNR=0
  PET=1

TYPE: STATE ERROR CHECK
  TYP=1

TEST NO: 1
  TNUM=1
  CALL ESET(GKCL)
  EE(1)=7
  CALL ESEXER(1,EE)
  CALL GINSK(WI1,SKDNR,ITNR,1,IPX,IPY,PET,EAREA(1),EAREA(2),
 :            EAREA(3),EAREA(4),BUFLN,LDR,DATREC)
  CALL ECHEKR(RESULT)
  IF (RESULT) NUM=NUM+1
  IF(.NOT.RESULT) CALL EREP(RESULT)

TEST NO: 2
  TNUM=2
  CALL ESET(GKOP)
  EE(1)=7
  CALL ESEXER(1,EE)
  CALL GINSK(WI1,SKDNR,ITNR,1,IPX,IPY,PET,EAREA(1),EAREA(2),
 :            EAREA(3),EAREA(4),BUFLN,LDR,DATREC)
  CALL ECHEKR(RESULT)
  IF (RESULT) NUM=NUM+1
  IF(.NOT.RESULT) CALL EREP(RESULT)

TEST NO: 3
  TNUM=3
  CALL ESET(WSOP)
  IF(.NOT.(WMI.EQ.WMO.AND.WISS.EQ.WOUT.AND.WMI.EQ.WOUT)) THEN
    EE(1)=38
```

APPENDIX F

LISTINGS FROM OPERATOR INTERFACE TESTS

```
        ANRY(I)=ANLY(I)
100   CONTINUE
C
C     SCALE DATA TO THE RANGE [0.0,1.0]
      DO 200 I=1,9
        WUX2(I)=0.1*WUX2(I)
        WUY2(I)=WUY2(I)*0.1
        WLX2(I)=0.1*WLX2(I)
        WLY2(I)=WLY2(I)*0.1
        ANRX(I)=0.1*ANRX(I)
        ANRY(I)=ANRY(I)*0.1
        WUX1(I)=0.1*WUX1(I)
        WUY1(I)=WUY1(I)*0.1
        WLX1(I)=0.1*WLX1(I)
        WLY1(I)=WLY1(I)*0.1
        ANLX(I)=0.1*ANLX(I)
        ANLY(I)=ANLY(I)*0.1
200   CONTINUE
      DO 300 I=1,12
         BODYX(I)=BODYX(I)*0.1
         BODYY(I)=BODYY(I)*0.1
300   CONTINUE
      ENDIF
      REPEAT=.TRUE.
C
      CALL GSLN(2)
      CALL GSPLCI(1)
      CALL GPL(9,ANLX,ANLY)
      CALL GPL(9,ANRX,ANRY)
C
      CALL GSFASI(1)
      CALL GSFAIS(1)
      CALL GSFACI(2)
      CALL GFA(12,BODYX,BODYY)
C
      CALL GSFAIS(3)
      call gsfasi(-2)
      CALL GSFACI(3)
      CALL GFA(9,WUX1,WUY1)
      CALL GFA(9,WUX2,WUY2)
      call gsfasi(-1)
      CALL GSFACI(4)
```

Section 4.2.1
Number 3

```
UPDATE WORKSTATION AND PROMPT FOR OPERATOR INTERACTION

CALL GUWK(WKID,0)
CALL OTESTR(WKID,1,0)

CLOSE AND RE-OPEN WORKSTATION TO CLEAR SCREEN

CALL GDAWK(WKID)
CALL GCLWK(WKID)
CALL GOPWK(WKID,WKCON,WKTY)
CALL GACWK(WKID)

*************************************************************

FRAME 1 - APPEARANCE OF OUTPUT PRIMITIVES

*************************************************************

DRAW HEADING FOR FRAME

CALL GSCHH(0.025)
CALL OBORDR
CALL GTX(.1,.95,'TEST OPOA01 : FRAME 1')
CALL GTX(.1,.9,'APPEARANCE OF OUTPUT PRIMITIVES')

DRAW EACH OUTPUT PRIMITIVE UNDER DEFAULT SETTINGS OF
ATTRIBUTES (OTHER THAN CHARACTER HEIGHT) AND TRANSFORMATIONS

THE PICTURE DISPLAYS A SCENE INVOLVING A SHIP AT SEA
A POLYLINE IS USED TO DRAW THE SHIP'S SAIL, POLYMARKER TO
DISPLAY STARS IN THE SKY, TEXT TO DRAW 'GKS' ON THE SAIL,
FILL AREA TO DRAW THE SHIP'S HULL AND CELL ARRAY TO DRAW
THE SHIP'S FLAG

CALL GPL(N1,X1,Y1)
CALL GPM(N2,X2,Y2)
CALL GTX(X0,Y0,TEXT)
CALL GFA(N3,X3,Y3)
CALL GCA(XL,YB,XR,YT,7,5,1,1,7,5,COLIA)

CHECKPOINT 2
UPDATE WORKSTATION AND PROMPT FOR OPERATOR INTERACTION
```

Program OP0A01

```
Function   2  GOPWK
Error   -106  Internal error in GERLOG
Function   4  GACWK
Error   -  6  GKS not in proper state: GKS shall be either in the state
              WSOP or be in the state WSAC
Function  12  GPL
Error   -  5  GKS not in proper state: GKS shall be either in the state
              WSAC or in the state SGOP
Function  14  GTX
Error   -  5  GKS not in proper state: GKS shall be either in the state
              WSAC or in the state SGOP
Function  14  GTX
Error   -  5  GKS not in proper state: GKS shall be either in the state
              WSAC or in the state SGOP
Function  12  GPL
Error   -  5  GKS not in proper state: GKS shall be either in the state
              WSAC or in the state SGOP
Function  13  GPM
Error   -  5  GKS not in proper state: GKS shall be either in the state
              WSAC or in the state SGOP
Function  14  GTX
Error   -  5  GKS not in proper state: GKS shall be either in the state
              WSAC or in the state SGOP
Function  15  GFA
Error   -  5  GKS not in proper state: GKS shall be either in the state
              WSAC or in the state SGOP
Function  16  GCA
Error   -  5  GKS not in proper state: GKS shall be either in the state
              WSAC or in the state SGOP
Function   8  GUWK
Error   -  7  GKS not in proper state: GKS shall be in one of the states
              WSOP, WSAC or SGOP
Function  12  GPL
Error   -  5  GKS not in proper state: GKS shall be either in the state
              WSAC or in the state SGOP
Function  14  GTX
Error   -  5  GKS not in proper state: GKS shall be either in the state
              WSAC or in the state SGOP
Function   8  GUWK
Error   -  7  GKS not in proper state: GKS shall be in one of the states
              WSOP, WSAC or SGOP
Function   5  GDAWK
```

Program OP0A01
GKS Error File

Program OP0A01
GKS Error File

```
Error    -  3  GKS not in proper state: GKS shall be in the state WSAC
Function    3  GCLWK
Error    -  7  GKS not in proper state: GKS shall be in one of the states
               WSOP, WSAC or SGOP
```

```
C
C     CLOSE WORKSTATION TO CLEAR SCREEN BEFORE TESTING
C     FUNCTION:  'CLEAR WORKSTATION'
C
      CALL GDAWK(WKID)
      CALL GCLWK(WKID)
      CALL GOPWK(WKID,WKCON,WKTY)
      CALL GACWK(WKID)
C
C     ***************************************************************
C
C     FRAME 1 - CLEAR WORKSTATION (ALWAYS)
C
C     ***************************************************************
C
C     DRAW HEADING FOR FRAME AND DRAW POLYLINE
C
      CALL OBORDR
      CALL GSCHH(0.025)
      CALL GTX(.1,.95,'TEST OPOA02 : FRAME 1')
      CALL GTX(.1,.9,'CLEAR WORKSTATION (ALWAYS)')
      CALL GPL(9,XC,YC)
C
C     CHECKPOINT 2 - PROMPT FOR OPERATOR INTERACTION
C
      CALL GUWK(WKID,0)
      CALL OTESTR(WKID,2,0)
C
C     CLEAR WORKSTATION - CONTROL FLAG = ALWAYS
C
      CALL GCLRWK(WKID,1)
C
C     CHECKPOINT 3 - PROMPT FOR OPERATOR ACTION
C
      CALL OTESTR(WKID,3,1)
C
C     ***************************************************************
C
C     FRAME 2 - CLEAR WORKSTATION (CONDITIONALLY)
C
C     ***************************************************************
C
```

Program OPOA02

Program OP0A02
GKS Error File

```
Function   2  GOPWK
Error   -106  Internal error in GERLOG
Function   4  GACWK
Error   -  6  GKS not in proper state: GKS shall be either in the state
              WSOP or be in the state WSAC
Function  12  GPL
Error   -  5  GKS not in proper state: GKS shall be either in the state
              WSAC or in the state SGOP
Function  14  GTX
Error   -  5  GKS not in proper state: GKS shall be either in the state
              WSAC or in the state SGOP
Function  14  GTX
Error   -  5  GKS not in proper state: GKS shall be either in the state
              WSAC or in the state SGOP
Function  12  GPL
Error   -  5  GKS not in proper state: GKS shall be either in the state
              WSAC or in the state SGOP
Function   8  GUWK
Error   -  7  GKS not in proper state: GKS shall be in one of the states
              WSOP, WSAC or SGOP
Function  12  GPL
Error   -  5  GKS not in proper state: GKS shall be either in the state
              WSAC or in the state SGOP
Function  14  GTX
Error   -  5  GKS not in proper state: GKS shall be either in the state
              WSAC or in the state SGOP
Function   8  GUWK
Error   -  7  GKS not in proper state: GKS shall be in one of the states
              WSOP, WSAC or SGOP
Function   6  GCLRWK
Error   -  6  GKS not in proper state: GKS shall be either in the state
              WSOP or be in the state WSAC
Function  14  GTX
Error   -  5  GKS not in proper state: GKS shall be either in the state
              WSAC or in the state SGOP
Function   8  GUWK
Error   -  7  GKS not in proper state: GKS shall be in one of the states
              WSOP, WSAC or SGOP
Function  12  GPL
Error   -  5  GKS not in proper state: GKS shall be either in the state
              WSAC or in the state SGOP
Function  14  GTX
```

```
Error    -  5  GKS not in proper state: GKS shall be either in the state
               WSAC or in the state SGOP
Function  14  GTX
Error    -  5  GKS not in proper state: GKS shall be either in the state
               WSAC or in the state SGOP
Function  12  GPL
Error    -  5  GKS not in proper state: GKS shall be either in the state
               WSAC or in the state SGOP
Function   8  GUWK
Error    -  7  GKS not in proper state: GKS shall be in one of the states
               WSOP, WSAC or SGOP
Function  12  GPL
Error    -  5  GKS not in proper state: GKS shall be either in the state
               WSAC or in the state SGOP
Function  14  GTX
Error    -  5  GKS not in proper state: GKS shall be either in the state
               WSAC or in the state SGOP
Function   8  GUWK
Error    -  7  GKS not in proper state: GKS shall be in one of the states
               WSOP, WSAC or SGOP
Function   6  GCLRWK
Error    -  6  GKS not in proper state: GKS shall be either in the state
               WSOP or be in the state WSAC
Function  14  GTX
Error    -  5  GKS not in proper state: GKS shall be either in the state
               WSAC or in the state SGOP
Function   8  GUWK
Error    -  7  GKS not in proper state: GKS shall be in one of the states
               WSOP, WSAC or SGOP
Function  12  GPL
Error    -  5  GKS not in proper state: GKS shall be either in the state
               WSAC or in the state SGOP
Function  14  GTX
Error    -  5  GKS not in proper state: GKS shall be either in the state
               WSAC or in the state SGOP
Function  14  GTX
Error    -  5  GKS not in proper state: GKS shall be either in the state
               WSAC or in the state SGOP
Function   8  GUWK
Error    -  7  GKS not in proper state: GKS shall be in one of the states
               WSOP, WSAC or SGOP
Function  12  GPL
```

Program OP0A02
GKS Error File

Program OP0A02
GKS Error File

```
Error    -  5  GKS not in proper state: GKS shall be either in the state
               WSAC or in the state SGOP
Function  14  GTX
Error    -  5  GKS not in proper state: GKS shall be either in the state
               WSAC or in the state SGOP
Function   8  GUWK
Error    -  7  GKS not in proper state: GKS shall be in one of the states
               WSOP, WSAC or SGOP
Function   5  GDAWK
Error    -  3  GKS not in proper state: GKS shall be in the state WSAC
Function   3  GCLWK
Error    -  7  GKS not in proper state: GKS shall be in one of the states
               WSOP, WSAC or SGOP
Function   2  GOPWK
Error    -106  Internal error in GERLOG
Function   4  GACWK
Error    -  6  GKS not in proper state: GKS shall be either in the state
               WSOP or be in the state WSAC
Function  14  GTX
Error    -  5  GKS not in proper state: GKS shall be either in the state
               WSAC or in the state SGOP
Function   8  GUWK
Error    -  7  GKS not in proper state: GKS shall be in one of the states
               WSOP, WSAC or SGOP
Function  12  GPL
Error    -  5  GKS not in proper state: GKS shall be either in the state
               WSAC or in the state SGOP
Function   8  GUWK
Error    -  7  GKS not in proper state: GKS shall be in one of the states
               WSOP, WSAC or SGOP
Function  14  GTX
Error    -  5  GKS not in proper state: GKS shall be either in the state
               WSAC or in the state SGOP
Function   8  GUWK
Error    -  7  GKS not in proper state: GKS shall be in one of the states
               WSOP, WSAC or SGOP
Function   5  GDAWK
Error    -  3  GKS not in proper state: GKS shall be in the state WSAC
Function   3  GCLWK
Error    -  7  GKS not in proper state: GKS shall be in one of the states
               WSOP, WSAC or SGOP
```

```
C - HATCH - OPTIONAL - BOTTOM RIGHT

      CALL GSFAIS(3)
      call gsfasi(-1)
      CALL OCONIC(.725,.3,.15,.1,90,270,2,FILL)

C ADD HOLLOW CHICK FOR VARIETY!

      CALL GSFAIS(0)
      CALL OCHICK

      CALL GTX(.725,.15,'HATCH')
      IF (HATCH) THEN
          CALL GTX(.725,.1,'(PRESENT)')

      ELSE
          CALL GTX(.725,.1,'(NOT PRESENT)')
      END IF

C CHECKPOINT 2 - PROMPT FOR USER INTERACTION

C FIRST RESET TEXT ALIGNMENT TO (LEFT,BASE)

      CALL GSTXAL(1,4)

      CALL GUWK(WKID,0)
      CALL OTESTR(WKID,2,0)

C ***********************************************************

C FRAME 2 - PATTERN AND HATCH STYLES

C ***********************************************************

      CALL GSCHH(.025)
      CALL GCLRWK(WKID,0)
      CALL OBORDR

C DRAW HEADING FOR FRAME

      CALL GTX(.1,.95,'TEST OPOA07 : FRAME 2')
```

Section 4.2.1
Number 3

Program OP0A07

```
C     FOR EACH SELECTED VALUE OF THE PATTERN INDEX
C
          CALL GSFAIS(2)
          J = 0
          DO 40 I = 1,6
              IF (RSET(I).NE.-1) THEN
                  J = J + 1
                  IF (I.EQ.6) J = 6
                  CALL GSFASI(RSET(I))
                  YO = .65 - FLOAT(J-1)*0.125
                  Y1 = YO + 0.05
                  WRITE (LABEL,'(I4)') RSET(I)
                  CALL GTX(0.05,Y1,LABEL)
                  CALL OTRIAN(.25,YO)
              END IF

   40     CONTINUE
      END IF
C
C     SELECT AND DISPLAY REPRESENTATIVE SET OF HATCH STYLES
C
      IF (NHS.EQ.0) THEN
C
C     HATCHING NOT SUPPORTED
C
          CALL GTX(.75,.825,'NO HATCHING')

      ELSE
C
C     HATCHING IS SUPPORTED
C
          CALL GTX(.8,.825,'HATCH')
C
C     CONSTRUCT A REPRESENTATIVE SET OF HATCH STYLES
C
          DO 50 I = 1,6
              RSET(I) = -1
   50     CONTINUE
C
          RSET(1) = 1
          MAXH = NHS
          IF (MAXH.EQ.1) GO TO 60
```

Section 4.2.1
Number 3

```
      RSET(5) = MAXH
      MIDH = (1+MAXH)/2
      IF (MIDH.EQ.1 .OR. MIDH.EQ.MAXH) GO TO 60
      RSET(3) = MIDH
      MIDMIN = (1+MIDH)/2
      IF (MIDMIN.NE.1 .AND. MIDMIN.NE.MIDH) RSET(2) = MIDMIN
      MIDMAX = (MIDH+MAXH)/2
      IF (MIDMAX.NE.MIDH .AND. MIDMAX.NE.MAXH) RSET(4) = MIDMAX
60    CONTINUE
      RSET(6) = MAXH + 1
C
C     SELECT INTERIOR STYLE HATCH AND DRAW A HATCHED TRIANGLE
C     FOR EACH OF THE SELECTED HATCH STYLES
C
      CALL GSFAIS(3)
      J = 0
      DO 70 I = 1,6
         IF (RSET(I).NE.-1) THEN
            J = J + 1
            IF (I.EQ.6) J = 6
            CALL GSFASI(RSET(I))
            Y0 = .65 - FLOAT(J-1)*0.125
            Y1 = Y0 + .05
            WRITE (LABEL,'(I4)') RSET(I)
            CALL GTX(0.55,Y1,LABEL)
            CALL GTRIAN(.80,Y0)
         END IF

70    CONTINUE
   END IF
C
   CALL GSLN(2)
   CALL GPL(2,XDASH,YDASH)
   CALL GSLN(1)
C
C     RESET FILL AREA STYLE INDEX
C
   CALL GSFASI(1)
C
C     CHECKPOINT 3 - PROMPT FOR USER INTERACTION
C
   CALL GUWK(WKID,0)
```

```
C     AND DISPLAY FILL AREA USING EQUIVALENT ATTRIBUTES
C     SET INDIVIDUALLY
C
      DO 100 I = 1,5
         CALL GQPFAR(WKTY,I,IERR,STYLE,STYLID,COLI)
         CALL GSFAIS(STYLE)
         CALL GSFASI(STYLID)
         CALL GSFACI(COLI)
         XO = 0.63
         YO = 0.65 - FLOAT(I-1)*0.15
         Y1 = YO + .05
         IF (ABS(STYLE).LT.10000) THEN
           WRITE (LABEL,'(I4)') ABS(STYLE)
           IF (STYLE.GE.0) THEN
              CALL GTX(.15,Y1,LABEL)
C
C     IMPLEMENTATION DEPENDENT FILL AREA INTERIOR INDEX
           ELSE
              CALL GTX(.15,Y1,'-'//LABEL)
           ENDIF
         ELSE
C
C     FIELD TOO BIG/SMALL FOR THE FILL AREA INTERIOR INDEX
           LABEL='***'
           CALL GTX(.15,Y1,LABEL)
         ENDIF
         WRITE (IXCHAR,'(I1)') abs(STYLID)
         CALL GTX(.32,Y1,IXCHAR)
         WRITE (LABEL,'(I3,1X)') COLI
         CALL GTX(.375,Y1,LABEL)
         CALL OTRIAN(XO,YO)
  100 CONTINUE
C
C     SET APPROPRIATE ASPECT SOURCE FLAGS FOR BUNDLED ATTRIBUTES
C
      ASF(11) = 0
      ASF(12) = 0
      ASF(13) = 0
      CALL GSASF(ASF)
C
C     NOW DRAW COMPARISON FILL AREA USING EACH BUNDLE
C
```

```
C ***********************************************************************
C
C     FRAME 1 - STORE ON METAFILE
C
C ***********************************************************************
C
      CALL GSCHH(0.025)
      CALL GCLRWK(WKID,0)
      CALL OBORDR
C
C     DRAW HEADING FOR FRAME
C
      CALL GTX(.1,.95,'TEST OP0A10: FRAME 1')
      CALL GTX(.1,.9,'STORE ON METAFILE')
C
C     DRAW PICTURE OF A FISH ON OUTPUT WORKSTATION
C
      CALL OFISH
C
C     CHECKPOINT 2 - PROMPT FOR OPERATOR INTERACTION
C
      CALL GUWK(WKID,0)
      CALL OTESTR(WKID,2,0)
C
C     DEACTIVATE AND CLOSE WORKSTATION
C
      CALL GDAWK(WKID)
      CALL GCLWK(WKID)
C
C     GET DETAILS OF METAFILE OUTPUT (MO) AND METAFILE INPUT (MI)
C     WORKSTATIONS
C
      CALL UWKVAL(MOID,MOCON,MOTY)
      CALL UWKVAL(MIID,MICON,MITY)
C
C     OPEN AND ACTIVATE METAFILE OUTPUT WORKSTATION, DRAW FISH
C     AND THEN DEACTIVATE AND CLOSE METAFILE OUTPUT
C
      CALL GOPWK(MOID,MOCON,MOTY)
      CALL GACWK(MOID)
C
      CALL OFISH
```

Program OP0A10

```
C       INDEX IS TOO LARGE
                CH2='**'
                CALL GTX(XST,.715,CH2)
          ENDIF
          NPI=NPLIX(I)
C
C       INQUIRE SET TEXT REPRESENTATION (DEFAULT ATTRIBUTES)
        CALL GQTXR(WKID,1,SET,IERR,FONT,PREC,CHXP,CHSP,COLIX)
        FT=FONT
        PC=PREC
        SP=CHSP
C
C       SET TEXT REPRESENTATION
C
        CALL GSTXR(WKID,NPI,FONT,PREC,CHXP,CHSP,COLIX)
C
C       DRAW TEXT (BUNDLED)
C
          CALL GSTXI(NPI)
          CALL GTX(XST,.65,STR1)
C
C       TAKING CARE OF NEGATIVE FONT NUMBER
          WRITE(CH3,'(I3)') ABS(FONT)
          IF (FONT.GT.0) THEN
            CALL GTX(.425,.65,CH3)
          ELSE
            CALL GTX(.425,.65,'-'//CH3)
          ENDIF
          WRITE(CH1,'(I1)') PREC
          CALL GTX(.55,.65,CH1)
          WRITE(CH3,'(F3.1)') CHXP
          CALL GTX(.625,.65,CH3)
          WRITE(CH3,'(F3.2)') CHSP
          CALL GTX(.75,.65,CH3)
          WRITE(CH1,'(I1)') COLIX
          CALL GTX(.875,.65,CH1)
C
C       SET TEXT FONT &  PRECISION INDIVIDUAL
C
          FONT=ASFONT(I)
          PREC=ASPREC(I)
          CALL GSTXFP(FONT,PREC)
```

Program OP1A03

```
      ASAF(7)=1
      CALL GSASF(ASAF)
      CALL GTX(XST,.5,STR1)

RESET TEXT FONT & PRECISION BUNDLED

      ASAF(7)=0
      CALL GSASF(ASAF)

      WRITE(CH3,'(I3)') FONT
      CALL GTX(.425,.5,CH3)
      WRITE(CH1,'(I1)') PREC
      CALL GTX(.55,.5,CH1)
      WRITE(CH3,'(F3.1)') CHXP
      CALL GTX(.625,.5,CH3)
      WRITE(CH3,'(F3.2)') CHSP
      CALL GTX(.75,.5,CH3)
      WRITE(CH1,'(I1)') COLIX
      CALL GTX(.875,.5,CH1)
      AX(1)=.45
      AX(2)=0.6
      AY(1)=.48
      AY(2)=AY(1)
      CALL GPL(2,AX,AY)
      FONT=FT
      PREC=PC

SET CHAR. EXPANSION FACTOR INDIVIDUAL

      CHXP=CHXP+0.5
      CALL GSCHXP(CHXP)
      ASAF(8)=1
      CALL GSASF(ASAF)
      CALL GTX(XST,.43,STR1)

RESET CHAR. EXPANSION FACTOR BUNDLED

      ASAF(8)=0
      CALL GSASF(ASAF)
      WRITE(CH3,'(I3)')FONT
      CALL GTX(.425,.43,CH3)
      WRITE(CH1,'(I1)') PREC
```

```
      CALL GTX(.55,.43,CH1)
      WRITE(CH3,'(F3.1)') CHXP
      CALL GTX(.625,.43,CH3)
      WRITE(CH3,'(F3.2)') CHSP
      CALL GTX(.75,.43,CH3)
      WRITE(CH1,'(I1)') COLIX
      CALL GTX(.875,.43,CH1)
      AX(1)=.625
      AX(2)=0.7
      AY(1)=.41
      AY(2)=AY(1)
      CALL GPL(2,AX,AY)
      CHXP=CHXP-0.5

SET CHAR. SPACING FACTOR INDIVIDUAL

      CHSP=ASCHSP(I)
      CALL GSCHSP(CHSP)
      ASAF(9)=1
      CALL GSASF(ASAF)
      CALL GTX(XST,.36,STR1)

RESET CHAR. SPACING FACTOR BUNDLED

      ASAF(9)=0
      CALL GSASF(ASAF)
      WRITE(CH3,'(I3)') FONT
      CALL GTX(.425,.36,CH3)
      WRITE(CH1,'(I1)') PREC
      CALL GTX(.55,.36,CH1)
      WRITE(CH3,'(F3.1)') CHXP
      CALL GTX(.625,.36,CH3)
      WRITE(CH3,'(F3.2)') CHSP
      CALL GTX(.75,.36,CH3)
      WRITE(CH1,'(I1)') COLIX
      CALL GTX(.875,.36,CH1)
      AX(1)=0.75
      AX(2)=0.825
      AY(1)=.34
      AY(2)=AY(1)
      CALL GPL(2,AX,AY)
      CHSP=SP
```

SET TEXT COLOUR INDIVIDUAL

```
COLIX=ASFCOL(I)
CALL GSTXCI(COLIX)
ASAF(10)=1
CALL GSASF(ASAF)
CALL GTX(XST,.29,STR1)
```

RESET TEXT COLOUR INDEX BUNDLED

```
ASAF(10)=0
CALL GSASF(ASAF)
WRITE(CH3,'(I3)') FONT
CALL GTX(.425,.29,CH3)
WRITE(CH1,'(I1)') PREC
CALL GTX(.55,.29,CH1)
WRITE(CH3,'(F3.1)') CHXP
CALL GTX(.625,.29,CH3)
WRITE(CH3,'(F3.2)') CHSP
CALL GTX(.75,.29,CH3)
WRITE(CH1,'(I1)') COLIX
CALL GTX(.875,.29,CH1)
AX(1)=0.85
AX(2)=AX(1)+.05
AY(1)=.27
AY(2)=AY(1)
CALL GPL(2,AX,AY)
```

SET TEXT ATTRIBUTES INDIVIDUAL

```
FONT=ASFONT(I)
PREC=ASPREC(I)
CHXP=CHXP+0.5
CHSP=ASCHSP(I)
ASAF(7)=1
ASAF(8)=1
ASAF(9)=1
ASAF(10)=1
CALL GSASF(ASAF)
CALL GTX(XST,.22,STR1)
```

```
DRAW ROW HEADINGS AND FILL AREA (BUNDLED)

YST=.68
YINC=.15
DO 50 I=1,6
  IF(NFAIX(I).LE.-1) GOTO 50
  IF(NFAIX(I).LT.10) THEN
          WRITE(CH1,'(I1)') NFAIX(I)
          CALL GTX(.02,YST,CH1)
  ELSE IF (NFAIX(I).LT.100) THEN
          WRITE(CH2,'(I2)') NFAIX(I)
          CALL GTX(.02,YST,CH2)
  ELSE IF (NFAIX(I).LT.1000) THEN
          WRITE(CH3,'(I3)') NFAIX(I)
          CALL GTX(.02,YST,CH3)
  ELSE IF (NFAIX(I).LT.10000) THEN
          WRITE(CH4,'(I4)') NFAIX(I)
          CALL GTX(.02,YST,CH4)
  ELSE

INDEX IS TOO LARGE
          CH2='**'
          CALL GTX(.02,YST,CH2)
  ENDIF
  INSTYL=INSTY(5-I+1)
  STYLIX=STYIX(5-I+1)
  COLIX=ASFCOL(5-I+1)
  IF (I.EQ.1.AND.INSTY(4).EQ.3) THEN
     INSTYL=3
     stylix=-1
  ENDIF

DRAW ATTRIBUTE VALUES

  IF (ABS(INSTYL).LT.1000) THEN
     WRITE (CH3,'(I3)') ABS(INSTYL)
     IF (INSTYL.GE. 0 ) THEN
         CALL GTX(.125,YST,CH3)

IMPLEMENTATION DEPENDENT FILL AREA INTERIOR INDEX
     ELSE
        CALL GTX(.125,YST,'-'//CH3)
```

Section 4.2.1
Number 3

Program OP1A04

```
DRAW TREE, MOUNTAIN, STARS AND LAKE AS FILL AREAS

CALL GSFAI(1)
CALL TREE(.7,.015,0.85)
CALL GSFAI(2)
CALL STAR(.2,.7,.85)
CALL GSFAI(3)
CALL LAKE(.02,025,.7)
CALL GSFAI(4)
CALL MOUNT(.38,.5,.6)

CHECKPOINT 5

CALL GUWK(WKID,0)
CALL OTESTR(WKID,5,1)
CALL GSCHH(0.05)

SET FILL AREA REPRESENTATION  -- CHANGE STYLE INDICES

DO 200 I=1,4

INQUIRE FILL AREA REPRESENTATION

  CALL GQFAR(WKID,I,TY,IERR,INSTYL,STYLIX,COLIX)

SET FILL AREA REPRESENTATION

  STYLIX=I

SET STYLE INDEX TO 3 IF HATCH INTERIOR STYLE IS PRESENT

  IF(INSTY(4).EQ.3) stylix=-3
  CALL GSFAR(WKID,I,INSTYL,STYLIX,COLIX)
200 CONTINUE

SELECT NORM. TRANSF. 3

CALL GSELNT(3)

SET CLIPPING INDICATOR 'OFF'

CALL GSCLIP(0)
```

Section 4.2.1
Number 3

```
      CALL GSFACI(COLIX)
C
C     SET INTERIOR STYLE INDIVIDUAL
C
      ASAF(11)=1
      CALL GSASF(ASAF)
      CALL TREE(.325,PY1,.5)
      WRITE(CH1,'(I1)') INSTYL
      CALL GTX(.39,PY1+0.02,'('//CH1//')')
C
C     RESET INTERIOR STYLE BUNDLED
C
      ASAF(11)=0
C
C     SET STYLE INDEX INDIVIDUAL
C
      ASAF(12)=1
      CALL GSASF(ASAF)
      CALL TREE(.48,PY1,.5)
      WRITE(CH1,'(I1)') STYLIX
      CALL GTX(.55,PY1+0.02,'('//CH1//')')
C
C     RESET STYLE INDEX BUNDLED
C
      ASAF(12)=0
C
C     SET COLOUR INDIVIDUAL
C
      ASAF(13)=1
      CALL GSASF(ASAF)
      CALL TREE(.65,PY1,.5)
C
C     SET STYLE INDEX INDIVIDUAL
C
      ASAF(11)=1
      ASAF(12)=1
      CALL GSASF(ASAF)
      CALL TREE(.825,PY1,.5)
      WRITE(CH1,'(I1)') COLIX
      CALL GTX(.725,PY1+0.02,'('//CH1//')')
C
C     SET FILL AREA ATTRIBUTES BUNDLED
```

1

```
ELSE
   CALL GTX(.355,.185,'(NOT PRESENT)')
ENDIF

CHECKPOINT 8

CALL GUWK(WKID,0)
CALL OTESTR(WKID,8,0)

SET FILL AREA ATTRIBUTES BUNDLED

  ASAF(4)=0
  ASAF(11)=0
  ASAF(12)=0
  ASAF(13)=0
  CALL GSASF(ASAF)

RESET PREDEFINED PATTERN REPRESENTATION

IF(NMX.GT.50.OR.MMX.GT.50) THEN

  CALL GSPAR(WKID,1,50,50,1,1,MMX,NMX,PRAY1)
ELSE

  CALL GSPAR(WKID,1,MMX,NMX,1,1,MMX,NMX,PRAY1)  ⇦ 2
ENDIF
IF(NMX1.GT.50.OR.MMX1.GT.50) THEN

  CALL GSPAR(WKID,2,50,50,1,1,MMX1,NMX1,PRAY2)
ELSE

  CALL GSPAR(WKID,2,MMX1,NMX1,1,1,MMX1,NMX1,PRAY2)
ENDIF

RETURN TO MESSAGE SECTION

GOTO 101

*****************************************************************

FRAME 5 - CHANGE OF PATTERN REPRESENTATION
```

```
ENDIF

SET FILL AREA ATTRIBUTES BUNDLED

  ASAF(4)=0
  ASAF(11)=0
  ASAF(12)=0
  ASAF(13)=0
  CALL GSASF(ASAF)

RESET PREDEFINED PATTERN REPRESENTATION

IF(NMX.GT.50.OR.MMX.GT.50) THEN
  CALL GSPAR(WKID,1,50,50,1,1,MMX,NMX,PRAY1)
ELSE
  CALL GSPAR(WKID,1,MMX,NMX,1,1,MMX,NMX,PRAY1)  2
ENDIF
IF(NMX1.GT.50.OR.MMX1.GT.50) THEN
    CALL GSPAR(WKID,2,50,50,1,1,MMX1,NMX1,PRAY2)
ELSE
    CALL GSPAR(WKID,2,MMX1,NMX1,1,1,MMX1,NMX1,PRAY2)
ENDIF

CALL GSELNT(0)

RETURN TO MESSAGE SECTION

GOTO 101

CLOSE AND DEACTIVATE....

90  CALL GDAWK(WKID)
    CALL GCLWK(WKID)

END OF TEST

CALL OTESTR(WKID,999,0)
CALL GCLKS

END
SUBROUTINE STAR(XO,YO,TIMES)
```

```
C
C     CLOSE SEGMENT
      CALL GCLSG
C
C     SET ASPECT SOURCE FLAGS 'INDIVIDUAL'
      CALL GSASF(IASF)
C
C     SET VIEWPORTS
C
      CALL GSVP(1,.2,.8,.2,.8)
      CALL GSELNT(1)
C
C     CREATE SEGMENT TO DRAW CIRCLE WITHIN A BOX
      CALL GCRSG(1)
      CALL OBORDR
C
C     SET FILL AREA INTERIOR STYLE 'HATCH' AND
C     COLOUR INDEX 2
      CALL GSFAIS(3)
      call gsfasi(-2)
      CALL GSFACI(2)
      CALL OCONIC(0.5,0.5,0.2,.2,0,360,10,1)
C
C     CLOSE SEGMENT
      CALL GCLSG
C
C     LET ASSIGN SEGMENT 1 HAS HIGHEST PRIORITY 1.0
      CALL GSSGP(1,1.0)
C
C     CREATE SEGMENT 2 AND ASSIGN A SPECIFIED PRIORITY SAY, 0.5
      CALL GCRSG(2)
      CALL GSSGP(2,0.5)
C     SET FILL AREA 'HATCH' AND COLOUR INDEX 1
      call gsfasi(-1)
      CALL GSFACI(1)
      CALL OCONIC(0.5,0.5,0.3,0.10,0,360,10,1)
C
C     CLOSE SEGMENT
      CALL GCLSG
C
C     OPERATOR PROMPT
      CALL OTESTR(WKID,15,0)
```

Section 4.2.1 Number 3

Section 4.2.1 Number 3

Program OP1A05

```
2100     FORMAT(/'TEST ENDED UNSUCESSFULLY IN FRAME 2')
C
      END

      SUBROUTINE HORSEH

C +   GKS CERTIFICATION .. OPERATOR TEST
C +   VERSION OF TEST SUITE ... 2.0 F77

C *** ***************************************************************
C *** **** DRAW PICTURE OF A HORSE HEAD.      [ OP1A05 ]  ***********
C *** ***************************************************************
      INTEGER N1,N2,N3,N4,N5,N6
      REAL EAR1X(14),EAR1Y(14),EAR2X(8),EAR2Y(8)
      REAL HEADX(31),HEADY(31),NECKX(27),NECKY(27)
      REAL EYEX(6),EYEY(6),NOSEX(6),NOSEY(6)
      integer scalef
      DATA N1,N2,N3,N4,N5,N6/6,6,31,14,8,27/

      DATA EAR1X/.74,.715,.7,.655,.61,.6,.61,.65,.7,.75,.78,.79,.77,
     :          .75/, EAR1Y/.975,.95,.9,.85,.8,.76,.74,.72,.7,.72,.86,
     :          .83,.87,.92/, EAR2X/.68,.66,.64,.6,.55,.515,.505,.55/,
     :     EAR2Y/.86,.945,.98,.95,.9,.85,.8,.76/, HEADX/.57,.515,.45,.4,
     :          .36,.29,.23,.19,.13,.1,.08,.1,.11,.15,.19,.225,.26,
     :          .27,.23,.25,.26,.3,.35,.36,.38,.42,.45,.48,.5,.54,
     :          .59/, HEADY/.78,.77,.74,.69,.65,.62,.61,.605,.575,.55,
     :          .51,.47,.45,.41,.4,.41,.415,.4,.36,.34,.33,.37,.42,
     :          .44,.43,.415,.43,.44,.45,.46,.465/,
     :     NECKX/.6,.64,.65,.64,.6,.55,.5,.44,.38,.35,.33,.34,.355,.9,
     :          .75,.95,.8,.97,.83,.97,.8,.98,.82,.96,.9,.84,.76/,
     :     NECKY/.65,.62,.575,.525,.47,.44,.4,.345,.3,.25,.185,.115,
     :          .04,.04,.075,.07,.195,.19,.34,.33,.49,.48,.59,.58,
     :          .66,.69,.72/, EYEX/.47,.44,.42,.44,.47,.49/,
     :     EYEY/.67,.67,.655,.64,.64,.655/, NOSEX/.1505,.15,.1495,.15,
     :          .1505,.151/, NOSEY/.525,.525,.515,.505,.505,.515/

      scalef=1
C     SCALE ALL DATA TO SPECIFIED SIZE
      DO 10 I=1,N1
         EYEX(I)=EYEX(I)*SCALEF+0.25
         EYEY(I)=EYEY(I)*SCALEF+0.25
```

System Development Corporation
A Burroughs Company

INTERCHANGE AND GRAPHICS STANDARDS FOR CALS APPLICATIONS

REVIEW AND ANALYSIS OF CALS-RELATED REQUIREMENTS FOR GRAPHICS STANDARDS

CONTRACT NO. 43NANB613292

TM-HU-900/000/00

31 JULY 1986

System Development Corporation
A Burroughs Company

INTERCHANGE & GRAPHICS STANDARDS FOR CALS APPLICATIONS

REVIEW AND ANALYSIS OF CALS-RELATED REQUIREMENTS FOR GRAPHICS STANDARDS

CONTRACT NO. 43NANB613292

TM-HU-900/000/00
31 JULY 1986

TABLE OF CONTENTS

TABLE OF CONTENTS

SECTION 1 - SUMMARY

The effort required under this contract was (1) to identify graphics interchange requirements for logistic technical information in order to establish a unified interface with industry for automated data exchange, and (2) to assess current, intermediate and long term capabilities for applying computer graphics standards to specific CALS applications, including identifying and prioritizing critical Research and Development issues. The following actions were taken to meet the requirements.

Applicable documents relating to the CALS program were identified, accumulated, and reviewed. In addition, fifty-four documents from surveyed projects, along with questionnaires and personal notes, were compiled into a CALS information database.

Participation in on-site interviews, in addition to feedback from NBS tours of other government facilities, was consolidated into the database of CALS information and used in the analysis.

Feedback was analyzed from questionnaires circulated to government personnel representing Automated Technical Manual Systems projects, Paperless Presentation and Maintenance Aids projects, and Automated Data Repositories and Product Definition Data projects. The NBS-sponsored CALS Workshop, held 24-25 June, 1986, provided further input from DOD projects and from industry, as well as an opportunity to further understand the requirements of the projects.

Active participation in the development of graphics standards provided a broad overview as well as detailed technical knowledge of the current state of and short term plans for those standards. Membership in the executive committee of the graphics standards organization led to an active role in the determination of future directions for graphics standards efforts and an insider's view of the advantages and shortcomings of the emerging standards.

Ongoing analysis of the resulting database led to the development of a high level reference model of CALS interchange requirements. From this model, analysis provided a suggested use of current and planned computer graphics standards, as well as obvious areas for further work in standardization.

Finally, two plans for handling the CALS graphics data interchange problem emerged: a short term plan, the standardization of a raster based CALS; and a long term plan, the inclusion of vector capabilities with the raster based CALS.

SECTION 2 - INTRODUCTION

This section contains the description of the purpose, the scope, and the approach to completing the contract.

2.1 PURPOSE

The Computer Aided Logistics Support (CALS) program is tasked with solving the problems resulting from the current paper-intensive logistics processes within the Department of Defense (DOD). The obvious solutions require automation of those processes, and a first step in the automation is a review of the diverse solutions currently in place within different government agencies. The results of the survey will then provide input into the overall design of a unified interface for automated data exchange. A necessary step in the development of the interface will be the provision for standardized input/output of the data. The media of computer graphics, the subject of this report, will be an essential element in that interface.

This contract is tasked with identifying computer graphics standards currently being developed, and areas for future standardization, that may be used to facilitate the automation desired by CALS. In this report, CALS applications' graphics interchange requirements are identified, and current, intermediate, and long term capabilities for applying graphics standards to these applications are defined. Critical Research and Development issues, resulting from the analysis of these requirements are detailed and prioritized.

2.2 SCOPE

The objective of CALS is to standardize interfaces between agencies, within agencies, and between government and industry. Specific areas of investigation for this effort are the preparation of technical manuals and the storage and retrieval of product definition data. Special emphasis is placed on graphics requirements for data interchange and review.

2.3 APPROACH

In order to fully analyze the requirements of existing CALS-related projects and to develop a knowledge of particular areas that could use graphics standards, a database of information about the projects and the general direction of the CALS program was necessary. The approach was to generate the database from existing documents, face-to-face interviews, and questionnaires, and then to perform an analysis of the data by categorizing the benefits of current graphics standards, categorizing the requirements for interchange of data in the CALS environment, developing an overall general reference model, and applying the categories of standards and requirements to the model. From this analysis, areas where current and planned graphics standards may be used, as well as areas where there is a need for further research and development, were identified.

SECTION 3 - FACTORS AND CRITERIA USED FOR ANALYSIS

Factors and criteria applied to the gathered data included information about applicable government projects currently in existence or planned that address the CALS problems. Also, the status of standardization of computer graphics functionality and the future plans for other areas of standardization played an important roll in determining the feasibility of future directions. Finally, inherent knowledge gained by experience with computer graphics applications, along with answers to the questionnaire sent to government projects, was used to determine critical areas of applicability of graphics standards.

3.1 EXISTING ONGOING EFFORTS

Current solutions to the logistics paper logjam were reviewed and analyzed for effectiveness and for possible melding into the overall CALS environment. Common requirements, problems and proposed solutions are evident from the detailed reports that emerged from responses to the questionnaires, face-to-face interviews, and analysis of written documentation.

3.2 EXISTING COMPUTER GRAPHICS STANDARDS

National and international standards in the field of computer graphics are emerging, with many diverse standards being developed. Section 4 provides a brief description of the graphics standards that are currently being generated and general points that relate to the CALS effort.

3.3 QUESTIONNAIRE AND OTHER QUESTIONS

An NBS-developed questionnaire was circulated to government facilities that met the criteria of paper-intensive logistics activities as well as to those that had begun to attempt to use logistic technical information in digital form. This questionnaire included a section dedicated to soliciting information about current computer graphics use and perceived future needs for computer graphics capabilities.

In addition to the questionnaire, certain other questions have been developed to be used during the evaluation and analysis of the data received.

a. Is there a possibility that the data on the screen may be changed? If so, is the change for 1) redisplay or hardcopy output, or 2) a permanent database change? This question helps determine whether modifications to an image on the screen are local to the terminal or must be fed back to a permanent database, such as through an IGES or CGM interface.

b. Is there a possible need for 1) presentation (report) output or 2) is the displayed data temporary (working data)? If presentation output is required, a CGI that interfaces to multiple device drivers would be useful.

c. Is there a need for auxillary data (non-graphic) to be saved with the picture? This question helps determine whether a IGES/PDES type of storage standard is needed rather than a purely graphic (picture) storage standard like CGM.

d. Is the system a highly interactive one? The answer to this question determines the need for complex I/O standardization, and may indicate dynamic picture modification (PHIGS).

e. Is there a need for diverse input devices? If so, a CGI graphics interface that provides many device drivers may be a suitable solution.

f. Will the required input/output devices always be available everywhere the system is to be accessed, or will downgraded I/O devices be sometimes used? Again, if there may be variations in I/O devices, the device independence of the graphics standards is a solution.

g. How much existing "old" software/hardware is there? The amount of existing software determines the feasibility of converting to standard interfaces. In cases of large amounts of existing software, a short term solution may require the implementation of a translation layer

between existing software and standard interfaces. If there is a lot of "old" hardware, conversely, there may be upgrades in the near future, suggesting that the prescribed upgrade time might be used to convert to the new graphics software interfaces while the system is already being modified.

h. Is there a need for three-dimensional manipulation? Is there a need for a three-dimensional model? These two questions determine different approaches to three-dimensional graphics software. If there is a requirement for three-dimensional rotation, but none for an internal model, then the three-dimensional extensions to GKS are indicated rather than the more complex system provided by PHIGS.

i. Are there plans for the use of personal computers for standalone use or as online devices? The use of personal computers in a system implies limited I/O capabilities, limited resolution to the screen, possibly no color capabilities, and limited memory availablity. Some graphics standards, such as PHIGS, define implementation facilities that would not fit in pc configurations.

j. How much Off-The-Shelf (OTS) software/hardware can be used? The use of OTS software/hardware is a cost effective measure, and standardization of interfaces leads to the development of such software and hardware. On the other hand, if unique requirements are imposed on the system, standard software and hardware may become burdensome, since translators would have to be written to and from each standard interface.

k. Is there a need for conversion of the output to other devices, such as hardcopy, large screen wall displays, or a speech synthesizer? If graphics output is intended only for a single vendor display screen, the need for a standard output is not so evident as when there are diverse types of output devices that may be required. However, even the most simple application meant for internal use by engineers may one day require the output be targetted to a large screen display for management review. The use of a CGI-based graphics interface facilitates the addition of other device drivers at a future time.

SECTION 4 - GRAPHICS STANDARDIZATION EFFORTS

This section contains a brief description of applicable current activities within computer graphics and other standardization areas. In general, each subject presented here represents a critical part of the CALS effort to provide automated data exchange, since each describes an effort to standardize one or more interfaces.

4.1 GRAPHICAL KERNEL SYSTEM (GKS)

The Graphical Kernel System (GKS) is already U.S. and an international standard. It defines two-dimensional graphics functions at the user interface level, providing the programmer with capabilities to create graphical output and accept graphical input from diverse graphical devices. In conjunction with the GKS functional specification standard, calling sequences to the GKS functions are also standardized for common programming languages (language bindings).

4.2 PROGRAMMER'S HIERARCHICAL INTERACTIVE GRAPHICS SYSTEMS (PHIGS)

The Programmer's Hierarchical Interactive Graphics System (PHIGS) is another user interface level graphics standard. It provides a richer set of capabilities, including support for CAD and process control applications, in a dynamic, highly interactive manner. An hierarchical database architecture is at the heart of the PHIGS design, with an accompanying ability to archive and recall from storage these structures on one or more workstations. The PHIGS standard is currently in development.

4.3 COMPUTER GRAPHICS INTERFACE (CGI)

The Computer Graphics Interface (CGI) standard, currently being defined, describes a standard method for exchanging device-independent data and controlling commands between applications programs and graphics devices. The CGI is designed to be a standard interface, on top of which graphics applications may be written. Conceptually, the CGI lies 'underneath' a GKS implementation, and talks directly to graphics devices.

4.4 COMPUTER GRAPHICS METAFILE (CGM)

The Computer Graphics Metafile (CGM) standard defines the data format for transfer and storage of graphics data and commands. The contents of the CGM represent snapshots of images generated by the applications program, that can be stored and re-displayed at other devices. Along with the CGM functional specification, there are three standard encodings for the CGM: character encoding, binary encoding, and clear-text (human readable) encoding.

4.5 IGES/PDES

The Initial Graphics Exchange Specification (IGES) and the Product Data Exchange Specification (PDES) are definitions of data exchange formats used primarily with Computer Aided Design (CAD) systems, to provide transportability of product data, and its associated display, from one CAD system to another. IGES is currently a standard; PDES is under development.

4.6 BINDINGS

To provide portable application software in general, programming language interfaces to functional standards (bindings) must be standardized. Specifically, for applications interface standards such as PHIGS and GKS, standard bindings to frequently used programming languages must be defined and utilized. As new computer graphics functional standards are developed, corresponding bindings will be generated. Similarly, when languages are updated, or new languages defined, each functional standard should be reviewed for possible new bindings.

4.7 CROSS LANGUAGE BINDINGS

Multiple-language facilities require access to a functional area such as graphics through more than one language interface. Standardization efforts just getting under way will help resolve this problem. The definition of a common language-independent procedure calling mechanism and of common data types will provide access in a portable, standard way. The standardization of these common elements, however, will involve a thorough survey of existing methods and data types, and must be accepted by the programming language community.

(Page 4-4 Blank)

4.8 COMMON STORAGE FORMATS

For graphic data exchange, common data storage formats provide a means for transferring a picture from one system to another. The systems do not need any commonality except for the software that reads/writes the picture storage. The computer graphics metafile standard (CGM), with new extensions currently being defined, provides such a standard data exchange for graphics data.

4.9 COMBINED TEXT AND GRAPHICS

New standards are being developed to define the combination of text and graphics. These standards must take advantage of existing graphics standards for I/O and for storage, if a more general use of the standards family, such as CALS requires, is to be achieved.

SECTION 5 - SURVEYED PROJECTS

A common reference model, shown in Figure 5-1, may be used to further understand the general interfaces that apply, and to demonstrate the areas where computer graphics standards may be indicated.

For each project and system surveyed, a brief description of the project is given, followed by observations and suggestions limited to utilization of applicable computer graphics standards, particularly in relation to the reference model. Key words and phrases are added to the general description in brackets (e.g., [raster format]) to highlight appropriate points.

5.1 PRINTING AND PUBLISHING SYSTEMS

5.1.1 AIPPS/600S

Warehouses full of pages [raster format] that need to go into a database are the inputs to the 600S system. Although they desire input to come from CAD systems [CAD], in vector format, for now "history has swamped us". They have three modes of output: laser printers, traditional typesetting equipment, and CRT screens [multiple device output].

Although it was felt by representatives of this project that the format the data arrived in, and what was required to be done with it, were not important, since they could do whatever was needed, there seems to be advantages to some consistent format for their data [image storage].

5.1.2 APPS

This is a pilot program that could tie in with EMPS (see below). It is a system that produces tech manuals (TM), tech bulletins (TB), changes to TM's and TB's, depot maintenance work requirements, and repair parts documents. Input data comes from word processors, though they report it is expensive to write translator programs; CAD systems, of which they have several, including AutoCad output in IGES format; typset reading optical character readers, a near-future plan, so they can read from the AIPPS database; and, in the future, DSREDS [diverse input sources].

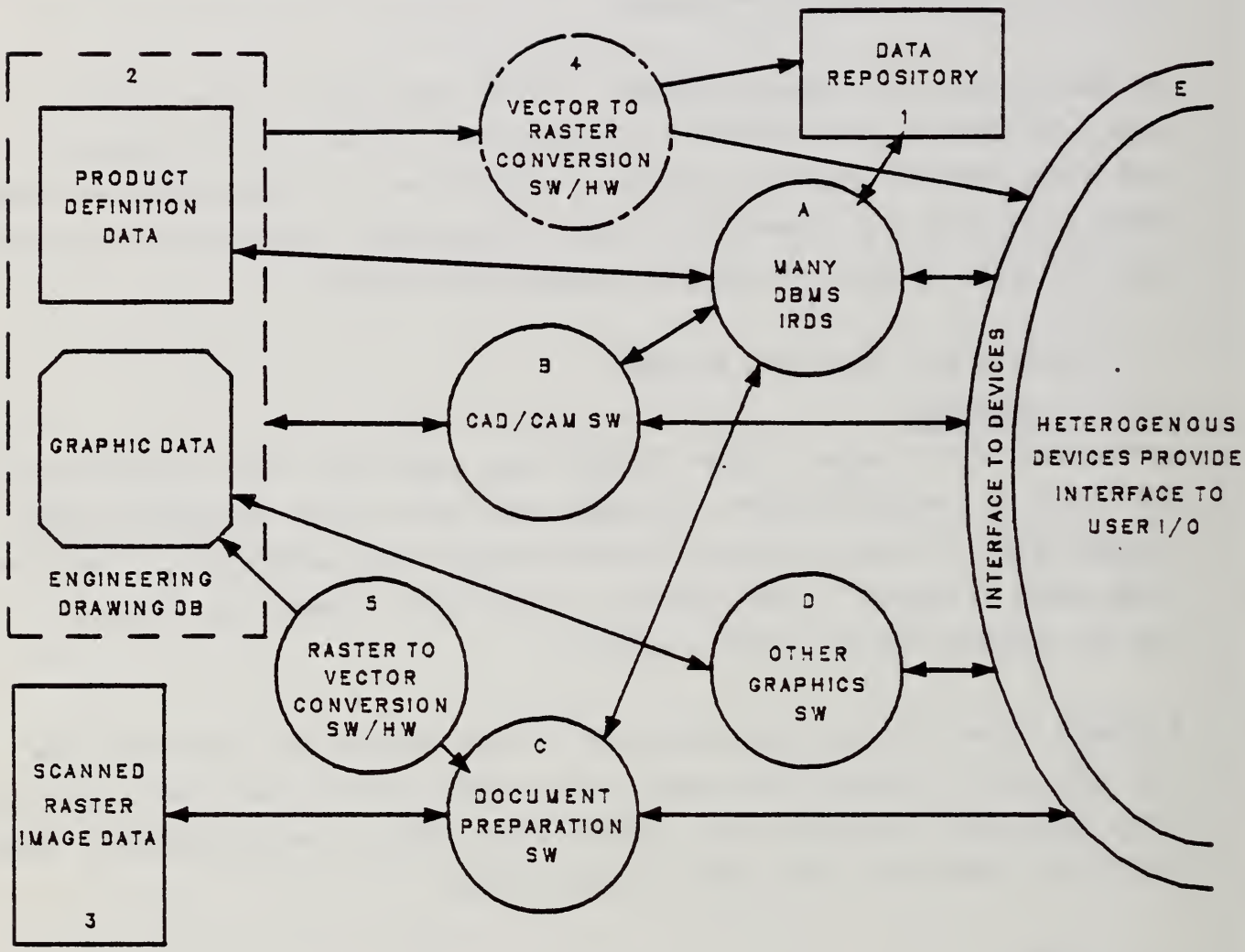


Figure 5-1. High Level Reference Model of CALS Graphics Requirements

Internal formats in APPS are used for 1) CAD data, 2) raster graphic manipulator output, 3) media converter output, 4) word processor output, and 5) mag tapes.

They modify the raster images almost all the time, but sometimes have to go all the way back to the CAD system to make the modifications [raster format] [vector format]. They feel that raster data formats are too large for on-line manipulation. They need something fast and structured. Although data compression techniques help, the compression formulas slow down storage and retrieval. Their raster storage requirements are enormous.

They recognize a common document maintenance problem: to store the original, a backup, and changes (both the original and a backup) [configuration management].

They use non-standard graphics languages and hardware, with no raster standard, no raster to vector conversion capability.

5.1.3 NAPPS

The Navy Automated Publishing and Printing System is used to develop training manuals for NTSC at Orlando and CNET at Pensacola. They want to be able to batch process their system, and want transportable, identical pages. Their system is pc-based, "not too esoteric", since they need blue collar composition.

Associated with NAPPS is the Logistics Technical Data Automation system, which is converting aperture cards to optical disk [raster format]. They have a huge installed data base, and can't afford republishing. The Navy is currently acquiring CAD/CAM capabilities, covering five different engineering disciplines, to provide support for the preparation of technical documentation [CAD]. Since they are providing services, they don't want to maintain the database.

5.1.4 NPODS

The Navy Print On Demand System (NPODS) has the problems that technical manuals, milars, vellums, and aperture cards [raster format] are input to print static manuals, which are only paper copies. Their solution is demand printing on a

laser printer. They feel there is no sense in offset printing. The original is used for mil specs, and standards. This system should be operational before October. A major problem they perceive is that data transfer rates for raster images are simply too slow. Their work "brings the 9700 to its knees".

This effort includes a contract that requires the conversion of aperture cards to optical disk [optical disks]. They estimate a cost of 15-17 cents per card. The drawings are up to E size. The need for optical disk standards was stated.

5.1.5 ATOS

A Wright Patterson Air Force Base system, the Automatic Technical Order System (ATOS) is currently in Phase 1, which is a computer-aided publications system, similar to APPS. Phase 2 will be taking data from paper, mag tape, and contractors, in a predefined format [diverse input sources]. However, it will be at least a year before this contract begins. They feel they "should be storing code, not images."

Phase 2 will include graphics workstations, with pan/zoom, cut and paste, 2-dimensional graphics capabilities, 1000 x 1000 resolution, 8 shades of gray, no color (but the database should be expandable for color), the capability for print-on-demand on a variety of printers, and input accepted from 100 different contractors. In addition to the workstations, there will be printers, both high and low speed, that can handle foldout pages and a minimum of 300 pixels per inch [multiple output devices].

Perceived ATOS long range needs are: a) an intelligent database where other AFLC systems can extract and supply TO information used to produce printed TO page [image storage]; 2) a system to convert raster pages to this database, since there are 20 million pages; and 3) ATOS database support of AFHRL enhanced presentation and interactive maintenance support [product data].

5.2 PAPERLESS PRESENTATION AND MAINTENANCE AIDS

5.2.1 MEIDS

The Militarized Electronic Information Delivery System (MEIDS), from the U.S. Army Ordnance Center at Aberdeen, is attempting to develop generic hardware to deliver text, graphics, and visuals [text and graphics]. Plans are for the "electronic page", with interactive capabilities, color, motion, and voice activation. They require text, graphics, and line drawings, with a distributed database and interaction between publications and communications [diverse input sources] [multiple output devices].

Input is scanned bit maps [raster storage], with some conversion for foldouts. Output is CRT display, and structured alphanumeric data. They also feel that the bit map approach requires HUGE storage capacity, but they don't want to base their design on the breakthrough technology of current raster to vector conversion systems. They decry the lack of a standard for optical disks [optical disks]. (ANSI is working on a large format, but MEIDS needs a small disk.) The cd rom access speed is too slow; MEIDS needs high speed interoperability.

MEIDS needs a graphics standard that permits economical porting to other hardware and for hardware upgrading [image storage] [portability]. They are interested in the long term maintenance of images [future enhancements]. They do not currently have 3-dimensional graphics needs, but mentioned that they may in the future. On-screen manipulation requirements are limited to pan, scroll, and zoom.

5.2.2 PEAM

The Personal Electronic Aid for Maintenance (PEAM) is a closed system, and there appears to be no obvious need for standards, other than human factors standards to help provide appropriate interfaces for the user of the system.

5.2.3 JPAPS

Another Army system, the Job Performance Aid Production System (JPAPS) is a two phase procurement. Common features of the current systems are text files, graphics files, and document design features [text and graphics]. Due to changing display technology, refresh speed and memory enhancements, they expect to be upgrading almost immediately [future enhancements]. They "need to get information on the screen in a hurry".

Phase 1 will involve [multiple output devices] [diverse input sources]:

- Research and development - techniques for displaying large schematics and line drawings - on small screens with low resolution (70 lines per inch). This includes conventions for overcoming the minimum resolution for printing 200 lines per inch.
- The identification and developing of authoring tools, new and existing.
- Analysis of common authoring requirements for existing technical information delivery systems.
- Applications of artificial intelligence. A first attempt will be for indexing and accessing information (tables and cross references).

Phase 2 will involve the development of software, some hardware, for proof of concept.

5.2.4 EMPS

The Electronic Maintenance and Publishing System (EMPS) is a project for transferring maintenance information for the Patriot system to video disk [optical disk]. They use read-only video disk and have no inhouse capability for mastering [future enhancements]. They are doing display-aided maintenance pictures, using a digital technique and high resolution screens. The system is IBM PC AT based, with software in Pascal and using dbase II.

They perceive the following needs for standards:

- Input: subject matter, formatting, content, composition, media (conventions for the input of data from contractors).
- Authoring: line drawings versus video, computer-generated text, interaction.
- Production: color use, presentation methods, use of motion and audio, formatting.
- Software: common language for production/authoring software. Currently programming is all being done in C; they anticipate the use of Lisp for AI related projects.

Storage is of display format, graphics, pictures, and text [image storage] [text and graphics]. Archiving requires input standards for contractor information, change control procedures, maintenance, and storage.

The optical disk needs standards in file design/formatting, interface loading and changing, and hierarchical data structures.

Standards for delivery include display size/resolution, color, input device design, textual criteria, design of portable terminals, database safeguards, database design, and human interface requirements [multiple output devices] [diverse input sources].

The system will be evaluated by USARMTE at Ft. Bliss in January 1987. Plans are to dovetail EMPS and APPS.

Their input is from Cromemco's AutoCad software. They want to be able to go back to the CAD system for major changes [CAD]. The maintenance function must modify almost every engineering drawing.

5.2.5 NTIPS

The Navy Technical Information Presentation System (NTIPS), being built by Northrop, involves:

- Computer selection of modular specs
- Automated preparation of contract packages
- Automated authoring.

They need optimal association of text and graphics for improved comprehensibility [text and graphics]. They don't want to tell the user at the authoring terminal how or what. The system is designed for shipboard delivery, with eventual voice input/output [multiple output devices] [diverse input sources]. It is to provide display manuals and training manuals. There is a three services study for the requirements for this system. (see CBAT below)

Technical issues to be resolved include:

- Standards for acquisition of data
- Repository of digitized data for print on demand
- Close association of text and graphics.

They consider a data-exchange standard very important [image storage].

The Navy has proposals for the content, style, format, and medium for data products. Also, they feel that test routines are absolutely necessary [validation].

5.2.6 CBAT

The Computer-Based Aids for Troubleshooting (CBAT) project is a 5 year Research and Development effort. It will be coordinated by IMIS (WPAFB) and NTIPS. It is required by CALS, to document the advantage of a paperless system over paper, and to gain user acceptance.

This system presents user-oriented technical information, in a six part approach:

1. Establish a workstation for developing portions of electronic presentation
2. Expand the database for apx-64 (fault detection)
3. Demonstrate in the field the feasibility of electronic delivery
4. Establish the feasibility of interfacing electronically presented information with in-system diagnostics
5. Use AI techniques to improve the organization
6. Define functional requirements.

The first phase, to establish a workstation, has just begun. It involves off-the-shelf hardware/software compatibility, with Unix-based tools. They will be using a Sun Microcomputer, and NTIPS and AFHRL software.

The second phase, to expand the database, includes an initial apx-64 electronic database, on loan from AFHRL [future enhancements]. It is an operational apx-64 and a pc, with fault pc boards, troubleshooting tools, and text and graphics [text and graphics].

5.2.6.1 CMAS/IMIS

An Air Force system, the Air Force Human Resources people are developing CMAS at Offett AFB. It uses voice and software programmable function keys [diverse input sources]. A novel idea is the use of peel-away graphics. In 1987, phase 3 begins. The biggest risk is identifying information flows (information engineering analysis). This system is to be used for new weapons sytems.

5.2.6.2 ITDS

The Improved Technical Data Systems (ITDS) is developed by Northrop. It is aimed toward solving the question: Where are we going to be in 1995? And then to identify, develop, and adopt the standards to take us there. One main problem is raster versus vector. They perceive that interactive electronic delivery, portable devices, and several output devices are part of the future [multiple output devices]. They recommend GKS for graphics.

Interfaces to be looked at are: 1) producer system and archive [image storage]; and 2) conversion coding of existing paper TO's and archive [raster format].

They expressed a need for fast graphics. IGES is currently used as a design tool [product data], and CGM could be used for portability, since they are output only [portability].

In the future, the engineering database from CAD/CAM systems will be the main input to technical publications [CAD]. The direct use of engineering drawings saves money (the use of high speed vector graphics), and provides quick retrieval of information (zoom and pan and color).

They are looking at 1280 x 1024 touch panel color displays for the future [future enhancements]. Currently they use a 512 x 512 flat panel display.

5.3 AUTOMATED DATA REPOSITORIES AND PRODUCT DEFINITION DATA

5.3.1 DSREDS/EDCARS

The Army's DSREDS and the Air Force's EDCARS system supply engineering, procurement, and drawing storage. For input, they use scan/digitized existing drawings [raster format], with interactive file retrieval. They use CAD support software for an engineering workstation that provides a look at the drawing in graphics so that changes can be made and stored. The images are stored on optical disk (juke box, read/write). They use 3D for manipulation of the images, with only 2D storage. There is a CAD/CAM interface to the disk, with a raster graphics interface to the QA terminal.

Conversion of aperture cards is a 6 - 8 month effort to store most of them.

They use IGES and CCITT G4 (raster) standards. They convert the 2D IGES file to raster and store 3D IGES by bit stream only. They are using IGES but only need 2D, non-modifiable images. So, they could be using CGM if an IGES to CGM interpreter was available.

5.3.2 TD/CMS

The TD/CMS is a system of 57,674 lines of undocumented code. Rockwell, in 1965 wrote the original code; in 1976 they rewrote it. However, they provided NO documentation. The code works, but no-one can modify or extend it. It is written in Cobol and Assembler for the IBM 4341, with 137 input data elements and 93 output reports. It is based on card input, and mag tape and card output which provides a pull tape for archival. In the future, they hope to provide hooks into DSREDS [portability].

The software trees down, but not up. They need this capability. They have already spent $600k and they still can't trace the code. A preliminary functional description of the redesign will be available this fall.

5.3.3 EDMICS

The Engineering Data Management Information and Control System (EDMICS) is the Navy version of DSREDS/EDCARS. It is a future product of the CALS initiative. It is the centerpiece of the Navy's answer to the CALS initiative. They want standards throughout. They want to represent images digitally [image storage]. They need database management standards for structure and query languages.

At the conceptual level, it will be the same as EDCARS/DSREDS. But is different in that it will specify more standards, and it will have more interfaces. The specification is in draft stages right now; it will be released to industry for comment soon.

They are looking at vector versus raster, with the initial opinion that they would like to get it 100% into vector mode [vector format]. "The Navy needs vector mode". There is a relationship to the aperture management (digitization) [raster format], with a proprietary compression algorithm. Some Navy locations use digital based optical systems for scanned drawings, and Portsmith, NH has the data management initiative. There will be an attempt to exchange files [portability] with DSREDS. INFODETICS is working on decompressed/recompressed format into CCITT.

5.3.4 PDDI/GMAP

The Geometric Modeling Applications Interface Program is a system to use product definition data. The GMAP builds on PDDI. They are attempting to replace engineering drawings [CAD] with an electronic interface between engineering and manufacturing. The results are a departure from IGES, since it contains non-geometric information (features, tolerances).

The end deliverable is to supply CAD feed to USAF/SA-ALC for IBIS (Integrated Blade Inspection System) and RFC (Retirement For Cause) inspection system.

They are working with PDES [product data] to define entities in generic form, not purely for the aerospace industry. 2D and 3D requirements coexist, but they feel it is not redundant to have both. They need a exchange format that is machine-readable, like IGES but smaller, not human readable [portability].

5.3.5 EDASRE

This DLA system, not yet procured, will be a part of the Navy procurement EDMICS. They receive 14 million aperture cards [raster format] annually. They may need more interfaces than EDMICS, to the Army, the Navy, the Air Force, and to industry. Some of their drawings need modifications. The DLA doesn't update the drawings; they normally receive revised drawings.

5.4 SUMMARY OF PROJECT NEEDS

5.4.1 Keywords

As may be noted by the keywords found in brackets in the above section, there were several common requirements and conditions that became apparent. This section will attempt to give a brief description of possible standards that could be used to satisfy these requirements and conditions. In some cases, there are no existing standardization efforts; these cases pinpoint areas where further development is suggested.

[CAD] - IGES with PHIGS or GKS to provide device independence

[configuration management] - procedures (to be developed)

[diverse input sources] - CGI, GKS, PHIGS (all device independent standards)

[future enhancements] - GKS, PHIGS

[image storage] - CGM, PHIGS Archive File, IGES, raster storage format (to be developed)

[multiple output devices] - CGI, GKS, PHIGS (all device independent standards)

[optical disks] - formatting standards (to be developed)

[product data] - PDES with GKS or CGI for device independence

[portability] - all standards in general

[raster format] - enhanced compression techniques, standardized format, raster-to-vector conversion technology (to be developed)

[text and graphics] - standards currently in development

[validation] - validation test suites and procedures (to be developed)

[vector format] - GKS, PHIGS

5.4.2 General Notes On The Technical Manuals Area

In general, the technical manual requirements are for a common DOD approach to SGML, with a committee to define the requirements of publishing.

For product definition data, there is a need for IGES to transport between CAD systems, for CGM to transport to publishing systems, and for a raster standard to store the print-on-demand images.

They need confidence in the standard products on the market. This implies strong, reliable validation procedures.

Configuration management must be a separate system from repository and publication systems. A data interchange standard is needed. A standard data access would be beneficial, such as a standard query language.

They need a raster-to-vector conversion capability. Product definition data is not needed to be carried along. They need a family of standards. They are looking at compatibility issues between IGES/PDES and the graphics standards. They want text and graphics capabilities in a single standard.

5.4.3 General Notes On The Data Repositories Area

There is an expressed need for textual data standards and database standards.

They agree there is no alternative to IGES as a product data definition standard. They stress the need for validation of IGES translators.

5.4.4 General Points On Graphics

There is a short range need for saving images in raster format, especially in those data repository areas where historical aperture cards are voluminous. However, for long range planning, they agree they want to transfer to an all-vector format. Identified problems with raster include configuration management problems for modifications and the possibilty of maintaining two or more versions (raster and vector from CAD systems), the size of the storage required, and the time to update the display with raster data.

They need validation for the standards' interfaces to provide security and portability.

There are some places where IGES is used that CGM could be used; however, in most cases both versions are needed.

(Page 5-16 Blank)

There is a continuing need for potential upgrade to future hardware; this implies the building of applications tools on top of CGI.

A single standard is needed for combined text and graphics.

SECTION 6 - REFERENCE MODEL

Figure 5-1 is an attempt to show, on a very high level, the interfaces described by the surveyed projects. The circles in the figures represent software systems, and the rectangles represent data formats.

The major software systems include database management systems (A), CAD systems (B), document preparation software (C), and other graphics applications software (D). In addition, there is usually some software interface to the diverse graphics output and input devices (E).

These software systems interface with various data formats, including the data repositories (1) which store image representations, the engineering drawing database (2) which is made up of product definition data and graphic data, and scanned raster image data (3) which is primarily data from aperture cards. Vector to raster conversion software/hardware (4) is used to transfer the CAD data to raster devices and to the data repository, and raster to vector conversion software/hardware (5) may be used to convert some of the CAD data output to vector document preparation format.

Within CALS, many format conversions are required. In almost all cases, format conversion in one direction can be performed automatically because the new format contains less information than the old format. For example, a 2D vector model contains less information than a 3D model; therefore, the 3D model can be automatically converted to a 2D vector model. Conversely, however, a 2D vector model cannot be converted to a 3D model without the addition of information which requires human interaction. Figure 6-1 illustrates this conversion hierarchy.

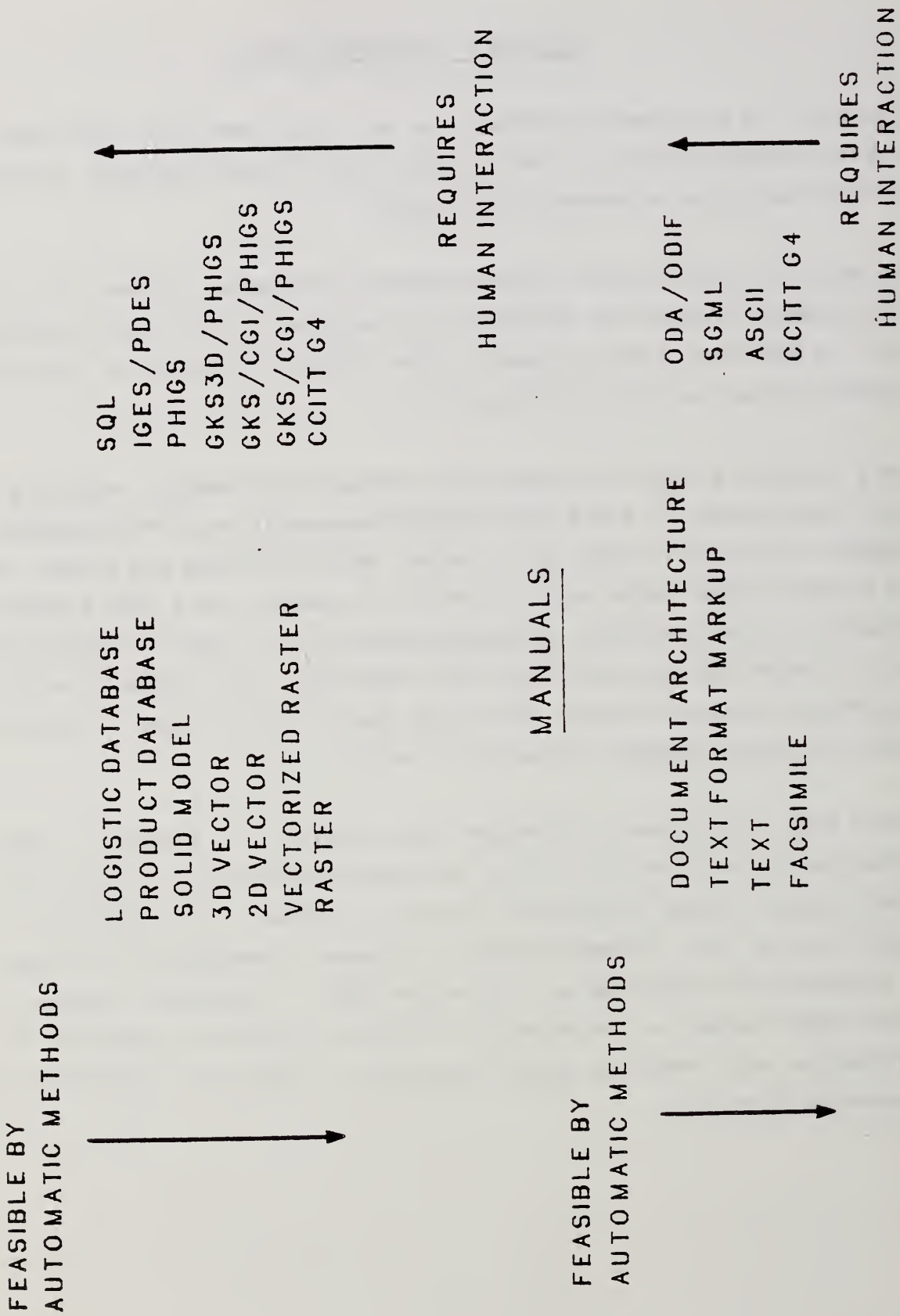


Figure 6-1. Conversion Hierarchy: Automatic and with Human Interaction

SECTION 7 - RESULTS OF ANALYSIS

This section details the results of the analysis previously described. The results are divided into two plans: a short term method for quickly implementing CALS philosophy and a long term plan for full implementation of the optimal CALS architecture.

7.1 SHORT TERM AND LONG TERM SOLUTIONS

From the resources investigated, two major graphics-related opinions emerged, in very clear and consistent statements throughout each project:

1. There is a tremendous backlog of data that must be accessed, manipulated, and output, currently in raster format. Much of this data is archived on aperture cards that must be digitally scanned. Data in this format will be a part of the CALS database forever.
2. Many of the current and future graphics data inputs will be originated at a CAD or CAE terminal that has capabilities for manipulating and storing 3D solid product models and/or an image in vector format. The database capabilities and engineering facilities provided by such systems, as well as the current trend towards lower cost, make their use highly desirable for production of the original drawings.

These two facts negate any attempt to determine a single, inclusive format for all CALS graphics data. Any solution must be able to handle both raster and vector types of images.

Vector images can be translated into raster format. In fact, in most cases, during display of the image on a graphics device, they are converted to a raster representation so that the hardware graphics engine can display the image. However, the reverse is not true, given the current state of the art. Although the command-and-parameter format of a vector image, such as "draw a line from point one to point two", can be fairly easily turned into a series

of on-off values for a raster display memory, it is much more difficult to recognize that same series of on-off points as a contiguous line. A major problem exists in simply checking the resulting output for validity and completeness. There are several industries that are attempting to resolve this problem. While raster to vector conversion is cost effective for drawing modification in some cases, currently there is no cost effective solution for document archival.

7.1.1 Short Term Solution

The least common denominator in terms of technical difficulty is the raster format of documents. The existence of raster scanned documents does not preclude the capture of hand-drawn images, CAD-generated documents or alphanumeric databases. Therefore, until standards are available to handle vector-formatted data, it is recommended that a raster based CALS system be standardized.

In such a system, CALS would store a raster facsimile of all DOD documents. Any document could then be located and transmitted anywhere in the world in seconds for display or printing. As standards for more complex data elements become available, CALS can be modified to store, retrieve, and integrate them. CALS could also store and retrieve binary files that do not conform to a standard, although it must be recognized that such files will be of limited value after time, due to inevitable modification of the systems that created them.

Raster image databases should be easier to manage than text, graphic, or inventory databases, since raster images contain fewer internal structures. This means that raster databases will not require distributed database management systems, enhancing the possibility of fast implementation.

Although much discussion by the surveyed projects has revolved around the problems of perceived slowness of display, and cost and amount of storage required by using only raster as the database format, current technological

trends are towards cheaper storage and faster decompression and raster display devices.

This short-term solution allows for the accommodation of current archival images as well as providing for the gradual development of the long-term solution described in the following section, without impacting the day-by-day performance of the projects.

7.1.2 Long Term Solution

CALS should support both raster and vector formatted data. The proposed long term solution is, while continuing to keep raster scanned documents in raster format, to migrate all incoming data to vector format. This means to encourage originals to be prepared on CAD systems, with a standard format for transferring and storing the vector version of each image. This format is already defined: it is the CGM standard format for the graphics operations, with a backup of IGES/PDES for the product definition data requirements.

In addition to using the established CGM and IGES/PDES standards for data storage, there is a need for standard interfaces to the target display devices. Many situations, from the soldier in the field attempting to upgrade or maintain some equipment, to the logistician in the office updating some information in a document, to management personnel briefing DOD Chiefs, require diverse devices for display and input to the data management programs that provide these capabilities. The emerging CGI standard, in addition to the GKS and PHIGS application-level standards, can be used optimally to provide the required device independence. A secondary benefit of using these standards is programmer portability for application software development/modification, and the cost benefits of increased availability of off-the-shelf hardware and software.

There will be, then, a standard storage base of images, in vector format if possible, that may be indexed, retrieved, and modified if necessary. The primary representation of the image, however, will remain the raster formatted version. The last step of every retrieval will be a rasterization step.

This solution provides both a quick, global access to documents and images, from the raster version of the image, and flexible storage of the same image in a format that is accepted from contractors in a standard way. The perceived configuration management problems with a dual representation method have actually been dealt with over the years as multiple versions of documents were created and maintained. Older versions of a document can be controlled in raster format while newer versions, rewritten or amended in alphanumeric or vector format, can be controlled in their new format. Controlling these multiple versions of documents is a capability that CALS must have whether the versions are in the same format or different formats.

As described above, reformatting such as raster to vector conversion, requires human input while the reverse vector to raster conversion can be done automatically. Reformatting requiring human input should be controlled as a new revision while automatic reformatting should be considered part of the output process. Because the output process can have several conversions with intermediate buffers, CALS must be designed to accommodate multiple automatically reformatted versions of a document. If automatic reformatting requires extensive computing, the intermediate results may be retained until the source document is changed.

CALS must also accommodate multiple identical copies of each document to provide spatial diversity for survivability of the combat readiness document base. CALS must have an integrated, on-line backup capability.

In addition, CALS must manage redundancy to speed distribution and conserve bandwidth. Copies of frequently used documents should be kept at remote locations to avoid frequent retransmission of recently received documents. To the user, these remote duplicates must appear transparent. That is, the user should have the same assurance of currency, and use exactly the same locate and access commands as the user would use for centrally located documents. The system should automatically select documents for remote duplicate storage and automatically choose the most accessible copy on demand. The foregoing

requires that CALS maintain virtual synchronization for multiple spatially distributed copies of documents.

It is of paramount importance that CALS have a well defined release mechanism for documents. The older manual systems could easily identify the release copy of a document and could easily require that all copies be made from that master copy. The CALS document release mechanism must identify the master copy of each released version of each document and assure that all copies are made from a released version.

The CALS document release mechanism will separate released documents from documents undergoing revision. While CALS may or may not contain preliminary or working copies of documents, such documents must be separated from the released documents CALS is intended to control and distribute. If CALS contains only released documents, the release mechanism becomes a CALS input requirement. If CALS contains released and unreleased versions of documents, the separation between the two types of documents will be extremely important.

CALS must maintain multiple versions of documents because each release describes parts for equipment that may still be in inventory. Because the cost of archiving CALS documents will probably be very low, because CALS documents will be well controlled and archived documents will be easily identifiable and not be confused with documents in day to day use, and because equipment previously in inventory may be recalled to active service (or may be in the hands of the enemy), CALS should maintain an archive of all versions of all documents entered into CALS.

SECTION 8 - CONCLUSION AND RECOMMENDATIONS

8.1 CONCLUSION

CALS must support raster images because almost all documentation either lacks an alphanumeric/vector form, or, if one is used, it is nonstandard. CALS must support standard alphanumeric and vector storage formats to facilitate full text searches and CAD input/modification. Choosing to remain with one or the other format exclusively would result in losing the capability to access archived raster images, or in losing the design and solid model information and the ability to search for text strings.

The raster-only short term solution is suggested, since it can be applied early, easily, and cheaply. In addition, a successful raster CALS system would almost certainly be able to inspire a budget for additional vector capabilities.

For the long term, however, CALS must also provide portability of software tools and ease of transfer of images and product data from one system to another. For this reason, CALS must accept documents in vector and alphanumeric formats as soon as standards for these formats are established.

8.2 RECOMMENDATIONS

CALS has been created to manage DOD's information. The best way to do this is to create a fully integrated information exchange system. To go electronic, the DOD must be able to display and print any document at any location, worldwide. Such a system would allow computer management and use of the DOD information database.

In addition, the DOD must begin to implement CALS quickly, in order to build on the interest and enthusiasm created so far. The complexity of a totally integrated system implies a leveled approach, with both short term and long term plans. The DOD must prioritize CALS features by desirability and technical difficulty.

The following subsections detail some areas where there is a clear need for future standardization or activity, in order to achieve the DOD objectives.

8.2.1 Applications Standards Interfaces

The interfaces between the applications standards should be clearly defined, with well documented descriptions of the functionality and data type correspondences across those interfaces.

Specific areas where such interfaces need to be determined, and the benefit gained in determining them, are:

- IGES/PDES to GKS/CGI/PHIGS - to allow for standard graphics interfaces to existing and new CAD/CAM systems.
- GKS/CGI/PHIGS to windows to UIMS - to provide standard interfaces from graphics applications to the user and to the specific user interface capabilities of devices.
- Various DBMS to UIMS and to GKS/CGI/PHIGS - to provide a standard method of database access from the graphics applications and from the user interface management software.

8.2.2 Raster Operations

Current computer graphics standards were developed with vector graphics capabilities as their primary concern. Recent technology has provided the industry with a proliferation of raster oriented functionality. In addition, the image processing world is melding with traditional graphics applications, particularly in the area of document preparation, a major CALS concern. Existing and new computer graphics standards must be reexamined in light of the prevailing requirements for image manipulation in combination with more traditional vector-oriented functionality.

Another capability that must be mentioned is that of converting raster images to a vector format that can be used by CAD systems. This is an area of intense

research and development, but no current systems have been designed for use in a document archive.

Any raster to vector conversion must be checked for errors. To estimate the cost of using raster to vector conversion, the level of error free conversion must be established and the number of operator hours required to achieve this level must be measured for each proposed system. While the cost of checking is usually less than the cost of redrawing and therefore very cost effective for drawing update, the cost of checking is still many times the cost of storing a compressed raster image of a drawing. Therefore, raster to vector conversion is suited to drawing modification, but not to drawing archiving. Similarly, OCR is cost effective for documents being edited, but is not effective for text documents being archived.

Because raster to vector conversion operates on a raster image, drawings stored in raster format can be connected to a vector format when extensive modifications are required. Simple modifications can be made in raster format by any one of several existing commercial systems.

8.2.3 Higher Level Application Software Standards

As application areas are identified, new software standards should be developed, built on top of the existing "family" of graphics and database standards. This further standardization is suggested to make available more off-the-shelf specialized software that uses the newly standardized application standards as a base. Some areas that may be approaching sufficient maturity for standardization are management information systems (MIS), user interface management systems (UIMS), business graphics capabilities, and database access routines (already in progress).

8.2.4 New Input/Output Facilities

As technology advances, the facilities provided for user interaction change and grow. The requirements for such facilities drive the need for a standardized access to them. Near future I/O devices that meet CALS require-

ments are speech synthesis for output to maintenance engineers, along with voice input for areas where other input methods are not feasible. Video is already being used for training and manual preparation; a standardized format for video would enhance future off-the-shelf software availability. Low cost eye tracking devices, with the device mounted on the display rather than on a head appliance, may prove ideal for cursor positioning on images.

8.2.5 Validation/Verification

For CALS to be successful, there must be validation 1) the risk of incompatibilities between contractors and DOD systems; 2) the risk of unreliable/noncompliant sources of data; and 3) the risk of rejection of data.

Although the validation/verification subject is listed last, it is probably the area of most concern for the CALS project. To meet the stated goals of providing cost-effective automation, with unified interfaces for automated data exchange, each part of each interface must meet certain requirements. These requirements are stated in the applicable standard for that interface. If the requirements are not met for even one connection in the system, the whole system fails. It is essential that some means be provided the government and industry to determine that the software interfaces comply with the stated standards.

TEXT

I.3 TEXTUAL STANDARDS

SPECIFIC TASKS

FY 86

1. Assess DoD needs for textual interchange standards:

   a. Identify text interchange requirements in CALS applications (e.g., technical publications, LSAR, reprocurement data).

   b. Recommend a set of textual interchange standards for specific DoD applications (e.g., SGML is a proposed standard for CALS interface with automated publishing systems). Assess specific near and long term benefits, limitations and impediments in adopting these standards for DoD use; the need for bridges between different textual standards; and alternative interim DoD approaches pending availability of the recommended standards and validation procedures.

   c. Develop a plan to expedite the development and implementation of textual standards for CALS based on the above findings.

   Deliverables:

   -- Report to CALS Steering Group on tasks a-b (preliminary report three months after go-ahead, final report at six months)

   -- Plan for textual standards area (outline three months after go-ahead, draft plan at six months; firm plan at eight months)

2. In parallel with task I.3.1 assess the following specific issues, and develop a paper on each:

   a. Quantification of the "overhead" inherent in the SGML approach for specific CALS applications; availability and expected benefits of SGML tools to facilitate preparation of information for delivery to DoD.

   b. Interface standards for text and graphics. Include an evaluation of current approaches to integrating CALS - relevant text and graphics standards.

   Deliverables:

   -- Separate issue papers (incrementally delivered within six months after go-ahead)

3. Accelerate textual standards development and validation efforts where needed to meet CALS schedule objectives:

a. Provide a plan for development and implementation of SGML validation procedures. Include criteria for DoD selection of validation projects. Identify schedule, resources, and recommended responsibilities for developing validation software.

Deliverables:

-- Quarterly status reports and a final technical report (eight months after go-ahead)

Tentative FY 87/88 Tasks

FY 87 and 88 tasks will be firmed up in the tactical plan delivered six months after FY 86 go-ahead. Tentative tasks include:

FY 87

1. Integration of SGML with the proposed standard office document architecture and interchange formats (ODA/ODIF).

2. Prototype the registration of a CALS document in SGML format with ANSI. That is, for a selected document type (e.g., an Air Force tech order), define the elements and mapping of elements to appropriate formatting primitives, and register with ANSI as an SGML-approved document. Evaluate the need for and recommend an approach for further SGML registration of DoD documents.

3. Specify a Navy DIF to SGML bridge (2-way).

4. Prepare a guidance document on the use of those subsets of SGML which are required by DoD.

5. Complete development of initial validation methodology. Prepare report describing initial methodology and begin evaluation of methodology.

6. Submit CALS defined document type to registration authority in X3V1.

FY 88

7. Continue to validate and enhance, if necessary, validation methodolcgy on prototype SGML systems.

8. Prepare a report describing validation methodology and procedures.

During FY 86-FY 88, participation will be required in ANSI X3V1 which is charged with the development of ODA/ODIF and SGML to ensure that enhancements to existing specifications reflect CALS requirements and to ensure registration of CALS documents by the SGML registration authority.

1.3 TEXTUAL STANDARDS

This section of the report addresses three specific tasks under 1.3 Textual Standards: Task 1.3.1 Assess DoD needs for textual interchange standards, Task 1.3.2 Assess the Following Specific Issues, and Develop a Paper on Each: (a. Quantification of SGML Overhead and b. Interface Standards for Text and Graphics), and Task 1.3.3 Accelerate textual standards development and validation efforts where needed to meet CALS schedule objectives.

1.3.1 ASSESS DoD NEEDS FOR TEXTUAL INTERCHANGE STANDARDS

1.3.1.1 Identify text interchange requirements in CALS applications (e.g., technical publications, LSAR, reprocurement data).

CALS applications have a number of requirements for text interchange, including:

- the need for a standardized way to exchange textual information (documents or parts of documents);

- the need for the exchanged documents to contain a variety of content, including character text, pictures, drawings, figures, and so on;

- the need to output the interchanged text on a variety of media (e.g., paper, CRT, laser printer, photocomposer);

- the need to pull together parts of documents prepared or processed separately, and, conversely, the need to distribute parts of documents for separate processing;

- the need for a standardized way to represent the appearance of the document (e.g., multi-column text, various fonts); and

- the need to use parts of the interchanged text in database applications.

(For another project, NBS prepared a paper on users requirements for document architecture and interchange format which can be made available to CALS.)

1.3.1.2. Recommend a set of textual interchange standards for specific DoD applications (e.g., SGML is a proposed standard for CALS interface with automated publishing systems). Assess specific near and long term benefits, limitations and impediments in adapting these standards for DoD use; the need for bridges between different textual standards; and alternative interim DoD approaches pending availability of the recommended standards and validation procedures.

There are a number of standards which can be used for text interchange. Two of these standards, SGML and ODA, are of primary interest to CALS. (In addition, there is a defacto text interchange standard, IBM's Document Content Architecture (DCA). The major limitation of DCA is its parochial nature.)

The SGML standard, a representation language for character text, has been recommended to CALS for technical publishing applications. SGML can be used for publishing in its broadest definition, from single medium conventional publishing to multi-media database publishing. SGML is used to describe whatever a user chooses to identify within a document. This results in an implicit document architecture, defined by the user. And it should be noted that while SGML is effective for representation of character text, there are some ambiguities in the way SGML handles other types of data. For example, SGML does not specify the format of graphics content; the user is free to describe this information any way he chooses outside the document.

On the other hand, ODA defines an explicit document architecture, including a capability for incorporating various content types in one document. This document architecture is the form of the information transmitted through a network. In fact, ODA relates only to the structure and format of a document in open interchange. This standard does not attempt to standardize any processes performed on the document either before or after interchange; therefore, the entry, editing, formatting and internal storage of the document may be different in each system. (Before interchange, however, the document will be translated into a standardized form (ODA). The recipient will translate the document to his own internal format and then process the document.)

To summarize, ODA addresses the problem of document interchange between unlike systems and SGML is a tool in a standardized set of tools for document management (everything from initial entry to final output).

The two standards are at different stages of development. SGML has become an international standard, is a planned national (ANSI) standard, and there will soon be many implementations available for CALS applications. The ODA work is a Draft International Standard and only prototype implementations of ODA exist. However, there are applications for the use of both the SGML and the ODA standards and it is likely that there are CALS applications for each. It is possible that future text interchange products will focus on accommodating the two techniques; that is, systems may be developed which can bridge SGML and ODA. This topic needs to be investigated further and a detailed report is a proposed deliverable for FY87.

1.3.1.3 Develop a plan to expedite the development and implementation of textual standards for CALS based on the above findings.

NBS staff are active participants in several national and international voluntary standards development efforts related to both SGML and ODA. These standardization efforts include the American National Standards accredited standards committee X3V1 (Text Processing: Office and Publishing Systems) and the International Organization for Standardization (ISO) Technical Committee 97 Subcommittee 18 (Text and Office Systems), which include the SGML and the ODA projects. The NBS role on voluntary standards committees, both as technical experts and unbiased (our viewpoint is not product-based) mediators, cannot be overstated. The plan to expedite the development of textual standards is based on this participation in voluntary standards activities.

With regard to the implementation of textual standards, NBS sponsors the NBS-OSI implementors workshops which we plan to expand to include implementations of textual standards in FY87.

1.3.2. **IN PARALLEL WITH TASK I.3.1 ASSESS THE FOLLOWING SPECIFIC ISSUES, AND DEVELOP A PAPER ON EACH:**

1.3.2.1 Quantification of the "overhead" inherent in the SGML approach for specific CALS applications; availability and expected benefits of SGML tools to facilitate preparation of information for delivery to DoD.

An issue paper per se has not been prepared; however, the following is a report of what is involved in quantifying the overhead in the SGML approach and of the availability of SGML tools for CALS applications.

A reliable quantification of the "overhead" inherent in the SGML approach for specific CALS applications is especially difficult and cannot be answered with any precise figure due to several important factors and considerations. These are listed below.

o In an SGML approach to document processing, there are three distinct aspects:

  - the design of the document type definition,
  - the actual marking up of the document, and
  - the outputting of the document to some medium.

  Since these processes are distinct, it is reasonable and correct to assume that, in the case of complex formatting requirements, the nonprocedural markup of an SGML document will be substantially easier than the procedural markup of the same document. In the case where the formatting requirements are simple but the logical structure of the document is complex, the markup process may require extra

effort.

- Overhead in the SGML approach must be relative to some other approach. Every document has overhead in the form of markup even though much of this may be transparent to the user. Without knowing what other approach might otherwise be used, it becomes difficult to state whether more or less overhead is involved using SGML.

- It must be recognized that there are several forms of overhead (e.g., the extra information that the operator was required to enter, the extra information stored with the document that was added by the text processing software.) Again, these types of overhead exist in all documents, regardless of the method used to create them, so there is no reason to believe that the SGML approach would be substantially different from other approaches.

- The extent to which markup minimization is employed, both in the document type definition and in the document elements, will have a great impact on the overhead involved - both in terms of operator effort and extra file information. This is especially true when data tag minimization is used!

- The amount of rework required for a document must also be included in estimating overhead. Since syntax directed text processing software may be available for SGML users, the amount of rework should be reduced and the effort spent in checking documents for completeness and correct structure should be virtually eliminated.

Since SGML is now an international standard (ISO 8879), tools should soon become available. Already there are syntax directed editors and formatting systems which are front-ended by SGML parsers and these are available both domestically and internationally. Although many of the systems are currently incomplete and need refinement, this will certainly come with time. Also, systems to tap the information in SGML documents and store that information in a database will permit data retrieval beyond the capabilities of systems that operate on procedurally marked up documents.

1.3.2.2 Interface standards for text and graphics. Include an evaluation of current approaches to integrating CALS - relevant text and graphics standards.

A discussion of the interface standards for text and graphics is included in the Graphics Interchange section of this report. It is important to note, however, that the textual standards described earlier each make use of graphics standards. Since the SGML standard allows any content to be defined by the user, the user could use an arbitrary graphics standard in defining graphics data. The ODA standard specifically mandates the use of particular graphics content architectures (part of the ODA

standard); these graphics content architectures are based on the Graphics Kernal System and the Computer Graphics Metafile (described in the Graphics Interchange section).

1.3.3. ACCELERATE TEXTUAL STANDARDS DEVELOPMENT AND VALIDATION EFFORTS WHERE NEEDED TO MEET CALS SCHEDULE OBJECTIVES:

1.3.3.1 Provide a plan for development and implementation of SGML validation procedures. Include criteria for DoD selection of validation projects. Identify schedule, resources, and recommended responsibilities for developing validation software.

A paper describing the framework for developing a validation package for SGML validation software has been prepared and was presented by NBS staff at the TechDoc '86 Conference. The paper, which defines a methodology for developing test suites to test conformance of SGML software to the SGML standard, is attached. Included in the paper are criteria for DoD selection of validation projects and schedule information.

SGML Parser Conformance Testing Methodology and Framework

Jim Heath
National Bureau of Standards
Institute for Computer Sciences and Technology

03 July, 1986

ABSTRACT

This document focuses on the development of a framework for testing SGML validating parsers. It examines the need to develop test suites, some of the particular problems associated with testing SGML parsers, approaches that were considered, a set of guidelines to be used in the actual test suite development, the actual organization of the test suite, and the test procedures. It is expected that the National Bureau of Standards will actually develop a set of validation procedures for SGML validating parsers to support the Computer Aided Logistics Support (CALS) initiative of the Department of Defense.

1. Introduction

The goal of SGML, like the goal of any standard, must be recognized as portability in some sense. In the case of SGML parsers, the focus is on document portability, i.e., the same document should not result in different interpretations when it is parsed on different systems. This objective cannot be completely achieved until parsers can be tested in order to determine whether they conform to the relevant standards.

Standard test suites should be developed for SGML parsers to be used by developers, users, or third-party testers; the test suites should be considered as evolving rather than static as they will be updated based on users' experience with parsers. The existence and acceptance of these test suites should lead to comparability and wide acceptance of test results produced by different examiners. This document to defines a methodology for developing conformance test suites.

2. Scope and Field of Application

This document defines a framework for the development of a test suite for SGML parsers and describes:

- definitions
- an overview of testing
- difficulties associated with testing SGML parsers
- approaches to testing
- organization of the test suite
- test procedures

NOTE: Unless explicitly stated otherwise, the term parser (as used in this document) will mean a validating SGML parser as defined in 6.3 of the SGML DIS. Although the standard also defines a conforming parser in section 6.3 of the SGML DIS, there is no requirement that it perform output of any kind and consequently its results cannot be validated.

The test methods described in this document address only the functional capacity of the parser - other features of an SGML system, such as user interface, performance, parser design, etc. are not considered.

Additionally, it should be recognized that "complete validation, implying absolute correctness, is presently infeasible with any sizeable program (DEUT82)." Only a small subset of the possible test cases can actually be presented to the validating parser. This limitation is not unique to the problem of validating SGML parsers but is characteristic of software testing in general; all one can do is submit a representative subset of the typically infinite number of possible inputs to the system under investigation and determine whether or not the results are in accord with the specifications for that system.

3. Definitions

3.1 Reference Model Definitions

This document is, in part, based upon the concepts developed in the ISO/DIS, Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML), November, 1985 and makes reference to the following terms defined in that standard:-

attribute definition
CDATA
connector
content model
element declaration
entity declaration
exceptions
exclusions
external identifier
feature
general entity
inclusions
NDATA
notation declaration
occurrence indicator
parameter entity
parser

Generally speaking, a parser is a program used to determine the underlying structure and content of some input object (file, document, etc.) More formally (in an SGML context), a parser checks that the tokens appearing in the input document occur in patterns that are permitted (by the rules of SGML and the description given by the document architect in the document type definition) and makes explicit the hierarchical structure of the incoming token stream by identifying which parts should be grouped together.

public identifier
ranked element
RCDATA
SGML declaration
system identifier
validating parser

4. Overview of Testing

Testing is a primary tool of software quality assurance and, in a broad context, encompasses not just the execution of the tests but also the design of test cases. The ideal goal would be to state, that after successful completion of the test suite, a software product is free from errors. This will not be possible in the general case and the best we can do is give the user confidence that the product is likely to perform as described. Although testing and debugging are often used interchangeably, we shall distinguish them by stating that the purpose of testing is to show the existence of errors while the purpose of debugging is to find the error or misconception and effect the appropriate corrections. Some of the characteristics of testing are:

1. Testing uses predefined inputs and expects a predictable set of outcomes. The only uncertainty is whether or not the software will execute the test correctly.

2. Testing is a demonstration of an error condition or the apparent correct processing of a document.

3. Testing can be designed and accomplished with ignorance of the internal program design.

4. Testing will involve treating the parser as a black box to which we provide a set of known inputs and from which we will take the output and compare it against the expected result.

5. Motivation for Developing Validation Procedures

The primary purpose of conformance testing is to establish whether the implementation being tested conforms to the specifications of the standard. We expect other benefits to derive from this:

1. It is generally agreed that the SGML standard is difficult to read and interpret - having a way to test their interpretations should provide a positive influence to developers of SGML parsers.

2. The fact that a parser has successfully completed a standard set of tests will improve its acceptability to users and give them confidence that they may process their important documents without misinterpretation.

3. As a result one 1 and 2 above, we expect that the acceptance of SGML as a standard will accelerate.

6. Difficulties Associated with Testing SGML Parsers

The testing of SGML parsers presents a particularly challenging situation because:

1. Parser output is loosely defined in the standard; also, the parser will be incapable of helping to diagnose its own mistakes (as contrasted to a compiler or interpreter which could perform some process and compare the outcome - in some cases - with a constant known value.)

2. Only minimal output is required by the parser - the parser is required only to report whether or not an error was encountered; no standard reporting form is required. Therefore, much of our evaluation of a parser's correct or incorrect handling of some function will be by inference, e.g., we cannot know that a parser has correctly interpreted an attribute value but we will infer that it has properly recognized the attribute value if it reports no error for a correct value and does report an error for incorrect values.

3. The tests cannot be modeled from the parser design because that will be unknown to the persons conducting the tests.

4. Various levels of implementation are likely since there are several functions in the standard which may not be useful to most users. The tests should be structured so that failure to process some rarely used function of the language will not disqualify a parser from further evaluation.

5. There is no requirement that an SGML parser continue after encountering an error, therefore, the number of exception test documents will be relatively large.

6. There are parts of the standard for which validation may not be possible, e.g., a validating parser which is not associated with any sort of formatting output process may fail to recognize 'record ends' which are caused by markup.

7. Finally, as with any complex computer application, complete validation is a goal that may never be attained. A failed test shows that a parser implementation <u>does not</u> conform to the standard; a successfully completed test shows only that it <u>may</u> conform. The completion of a series of well constructed tests establishes confidence that the software will perform as intended.

It is also important to keep in mind that this document addresses testing parsers for conformance, not testing documents for conformance - the distinction is important. In the former,

conformance is a matter of syntax; if a document has been constructed according to the rules of SGML, it is compliant. Furthermore, we can determine a document's conformance or nonconformance by inspection.

In contrast to document conformance which is described structurally, parser conformance is described functionally. The essential requirement for a parser is that it accept as input, any document and inform the user if it cannot determine its underlying structure and content in accordance with the rules of SGML.

7. Approaches to Testing

Testing techniques are as varied as programs are varied and there is no single best method. Instead, some combination is usually most effective. A few of the primary approaches are described below:

7.1 Path Testing

A path, as used in this approach, is some executable sequence of instructions through a routine. Path testing involves choosing input data such that enough paths are generated so that:

1. Every instruction in the routine is exercised at least once.

2. Every decision (branch or case statement) has been taken in each possible direction at least once (BEIZ84).

Path testing is highly regarded as the corner stone of testing; however, to know which paths to test requires knowledge of the parser's internal design which will generally be unavailable to the tester. Also, since each parser will be implemented differently, we would be forced to treat each one individually. For these reasons, path testing - despite its great value - cannot be used in developing validation procedures for SGML parsers and will not be further considered.

7.2 Transaction Flow Testing

Transaction flow testing provides another approach. From the user's point of view, a transaction is simply a unit of work - from a functional point of view, a transaction is a sequence of operations beginning with an input and resulting in one or more outputs. At the completion of a transaction, it is no longer in the system (except perhaps for some historical record). Examples of transactions are inquiries into reservation systems, withdrawals from automatic teller machines, etc. Although there are no clear counterparts to transaction processing in parsing an SGML document, we can consider some functions (e.g., defining entity references, defining content models, opening files associated with external entities) as transactions even though we may only be able to infer that they were processed correctly or not based on some later event, e.g., we will not know whether a content model in the document type definition was processed correctly until we encounter the element in the document). This modified form of transaction flow testing will be used frequently in developing the validation suite.

7.3 Syntax Checking

In syntax testing, the object under test is treated as a black

box which should accept valid inputs and reject invalid ones. The emphasis is not on what the program or system does with the inputs (and in the case of a parser we do not know with confidence what it does with the inputs) but rather on whether or not it correctly distinguishes valid from invalid inputs. This type of testing is highly applicable to SGML parsers (indeed, it forms the basis for defining validating parsers) and we will exploit it as much as possible. Syntax testing is used to demonstrate the following:

1. The system does not fail when subjected to bad inputs.

2. The system rejects all bad inputs and accepts all good inputs.

3. The system correctly process valid inputs.

7.3.1 Categories of Syntax Errors

(BEIZ83) defines eight categories of syntax errors:

1. High-level syntax errors: the strings have violations of the topmost level in a top-down BNF syntax specification.
2. Intermediate-level syntax errors: syntax errors at any level other than the top or bottom.
3. Field-syntax errors: Syntax errors associated with an individual field where a field is defined as a string of characters that has no subsidiary syntax specification other than the identification of characters that compose it. A field is the lowest level at which it is productive to think in terms of syntax testing.
4. Delimiter errors: Violation of the rules governing the placement and the type of characters that must appear as separators between fields.
5. Field-value errors: Not really syntax errors, but errors associated with the contents of a field.
6. Syntax-context errors: When the syntax of one field depends on values of other fields, there is a possibility of an interaction between a field value error and a syntax error. For example, when the contents of a control field dictate the syntax of subsequent fields.
7. Field-value correlation errors: Then contents of two or more fields are correlated by a functional relation between them. There is not full freedom in choosing their values. The value of one field is restricted by another field's values.
8. State-dependency errors: The permissible syntax and/or field values is conditional on the state of the system or the routine. For example, a command used for startup may not be allowed when the system is running.

7.3.2 Test Case Design

The basic strategy will be to create one error at a time while keeping all other parts of the input statement correct. A logical next step would be to consider double errors, triple errors, etc. but the number of test cases would increase exponentially so this is not feasible. Another and perhaps more viable approach would be to select compound cases that are likely to be reveal vulnerability in the parser but without knowledge of the parser design this becomes almost impossible. Initially, therefore, we will
test single error cases only. If experience reveals particularly troublesome combinations, they may be incorporated later.

7.3.2.1 Top, Intermediate, and Field-Level Syntax Errors

Again (BEIZ82) offers some help in selecting test cases; assume the topmost syntax level is defined as:

item := a | b | (c & d)

1. Do it wrong! Use an element which is correct at some lower syntax level but not the current one.
2. Invalid combination! For example, from the above definition use (c & b) rather than (c & d).
3. Don't do enough! Use (c) instead of (c & d).
4. Don't do anything! Many systems fail when the input is null; also, other problems (apparently unrelated) may be revealed.
5. Do too much! Use (a & b) instead of just (a).

Concentrate on only one level at a time, trying to keep the levels above and below as correct as possible.

7.3.2.2 Delimiter Errors

Delimiters are used to separate fields or parameters and the problems associated with them may provide a rich source of test cases. Some cases to include would be (BEIZ83):

1. Missing Delimiter! This causes the apparent merging of two fields.
2. Wrong Delimiter! For example, use single quote for a parameter separator, etc.
3. Not a Delimiter! Use some character or string which is not a delimiter where a delimiter should exist.
4. Too Many delimiters! Perhaps the system becomes confused.
5. Paired Delimiters! There are lots of possibilities here including nesting, unpaired delimiters, and compound errors such as ((()(())).
6. Tolerant Delimiters! The delimiter may be optional or more than one form may be acceptable.

7.3.2.3 Field-Value Errors

In some cases, values are associated with fields; and the possible entries for these values should be checked.

1. Boundary Values! Good choices would include minimum - 1, minimum, minimum + 1, a reasonable value, maximum - 1, maximum, maximum + 1, very much below minimum, very much above maximum.
2. Excluded Values! Test for values which should be excluded (if any).
3. Troublesome values! For numeric values check for values surrounding powers of 2.
4. Type Changes and Conversions! If a field value should be encoded as a string of ASCII digits, put in some nondigit, are leading +/- signs allowed, etc.

7.3.2.4 Context-Dependent Syntax Errors

Sometimes variations of syntax may be allowed depending on context; in the case of SGML, a good example would be a contextually required start tag.

7.3.2.5 Correlated Field Values

7.3.2.6 State-Dependency Errors

The format of a statement may be acceptable at one time but not another depending on the system's state. In the case of SGML parsers, it might be permissible for a paragraph to occur inside a chapter, but not inside a figure.

7.4 Logic Based Testing

This approach to testing assumes that the tester has access to the rules that formed the specification - usually something like a decision table. The idea is that the same rules that were used for the design may also be used for testing. Generally, this information does not exist in any usable form for SGML parsers so we will discount the technique from further consideration.

8. Organization of the Test Suite

The following are some general guidelines that should be kept in mind in designing the test suite.

8.1 Avoiding Reliance on Untested Functions

One of the significant problems confronting the test designer is to find some organizing principle, i.e., a natural way of sequencing the tests. One such approach would be to test the functions in the order in which they appear in the standard; the fundamental problem with this strategy is that the organization of the standard does not lend itself to the bottom up testing required to logically test a parser. A function cannot be effectively used in testing another related function unless it has previously been tested, e.g., a comment within a declaration cannot be used until it is known that the parser will recognize a comment and the resolution of an entity cannot be tested until it is known that the parser can process an entity declaration. Using this approach lessens the possibility that the function being tested could wrongly pass the test because of a flaw in the implementation of a function whose validity is being assumed. Also, if an error were reported by the parser, it would not be clear whether the true cause of the failure was the function under test or one of the untested functions being used.

8.2 Test All Individual Functions

The test suite will be subdivided into test groups with each group primarily constructed to test the general ability of a parser to correctly process functions of the standard. Test groups are in turn subdivided into test subgroups which test a parser's capacity to handle subsets of the function being tested. The test subgroups are built from the actual test cases which test the parser's capacity to handle specific instances of a function.

The scheme of testing only one function per group also helps to minimize the total number of tests since it eliminates the extreme growth in the number of tests which would result from testing all possible combinations. This approach is not without problems since it is likely there will be some level of interdependence between the processing of various functions. Where this is recognized, special test groups will exist to specifically test function combinations.

8.3 Minimize the Number of Tests

Ultimately, the input to any process must be viewed as a bit stream, therefore a process which accepts a two byte (assuming a byte is 8 bits) input could have 65536 distinct input possibilities. If a process accepts four byte inputs and

requires 100 microseconds to test each input, it would require almost 120 hours to check all inputs. Obviously this is not a reasonable for systems like SGML parsers where the inputs may be several orders of magnitude more complex. We must therefore carefully choose a very finite subset of the possible set of test cases and use this for our test suite.

8.4 Make the Tests Easy to Use and Their Results Easy to Interpret

It should be accepted that the tests will be executed in different environments by persons whose primary concern is determining the conformance of a parser, not in trying to understand how to perform the tests. Also, the tests should be structured, as much as possible, so that the results are easy to understand and interpret.

9. Test Procedures

The main focus of conformance testing will center on 'live' processing of a test document(s) by the parser being tested, i.e., documents - conforming and/or non-conforming - are given to the parser and a verdict is determined for its behavior. If the document is conforming, the parser should report no errors; if the document is non-conforming, the parser should note an error. An observer will analyze the outcome and if it is as expected will consider the test successfully completed, else he will consider it failed.

The testing will be organized in accordance with section 8.1 of this document, i.e., a bottom-up ordering will be followed. This implies that a complete document cannot be tested until it is known that a document type definition can be correctly processed. Since the document type definition is built from subordinate components, it is necessary to test them individually before testing entire type definitions. A logical ordering of the test sequence for these components is:

Comments
SGML Declaration
Parameter Entity Declarations
General Entity Declarations
Content Model Declarations
Marked Section Declarations
Exceptions Declarations
Attribute Declarations
Content Declarations
Complete Document Type Definitions

After it is determined that the parser can successfully process document type definitions, testing of complete documents can follow. Following the same bottom-up approach as used for the document type definition, testing would begin with the simplest SGML document and proceed in a logical sequence to the more complex up to the stated limits of the parser.

10. References

a. Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML), ISO/DIS 8879, November, 1985.

b. Revised Working Draft for OSI Conformance Testing Methodology and Framework, ISO TC 97/SC 21/WG 1, November, 1985.

c. NBS Minimal BASIC Test Programs - Version 2, User Manual, Volume 1 - Documentation, U.S. Department of Commerce, National Bureau of Standards, November, 1980.

e. Characteristics of Software Quality, Boehm, Brown, Kaspar, Lipow, Mac Leod, and Merritt, TRW Series of Software Technology, 1978.

f. Software System Testing and Quality Assurance, Beizer, Van Nostrand Reinhold Electrical/Computer Science and Engineering Series, 1984.

g. Computer Program Testing, Chandrasekaran and Radicchi, North-Holland Publishing Company, 1981.

h. Software Testing Techniques, Beizer, Van Nostrand Reinhold Electrical/Computer Science and Engineering Series, 1983.

i. Principles of Compiler Design, Aho and Ullman, Addison-Wesley Series in Computer Science and Information Processing, 1979.

DATABASE MANAGEMENT

II.1 DATABASE MANAGEMENT SOFTWARE AND STANDARDS

SPECIFIC TASKS

FY 86

1. Assess DoD needs for database management standards:

   a. Identify database characteristics in planned CALS applications (e.g., contractor design, manufacturing and logistic databases, LSAR databases, reprocurement data).

   b. Recommend a set of standards, methodologies and tools for DoD use in database design (including data modeling, data dictionaries, standard query languages and architectures for distributed databases), and additional software (language bindings, etc.) needed for CALS applications.

   c. Assess current, intermediate and long term capabilities to implement these standards to meet CALS distributed database needs. Identify and prioritize critical R&D issues.

   d. Recommend a strategy for use of a data dictionary in CALS planning.

   e. Develop a plan to expedite the development and implementation of database management software and standards for CALS based on the above findings.

   Deliverables:

   -- Report to CALS Steering Group on tasks a-d (preliminary report three months after go-ahead, final report at six months)

   -- Plan for the database standards area (outline three months after go-ahead, draft plan at six months, firm plan at eight months)

2. Accelerate database standards development and validation efforts where needed to meet CALS schedule objectives:

   * a. Provide a milestone plan for gaining acceptance of the standards recommended in task II.1.1.b.

   b. Develop a validation approach for data dictionaries.

   c. Develop a data dictionary shell for use in specific CALS demonstrations.

   * d. Develop preliminary functional specifications for data dictionary extensions to support needs identified in task II.1.1.b. This might include graphics, distributed databases and data modeling.

Deliverables:

-- Quarterly status reports and a final technical report (eight months after go-ahead)

3. In support of tasks II.1.1 and II.1.2, evaluate DoD demonstration programs and, where necessary, conduct additional experiments on NBS testbeds to resolve technical issues.

Deliverables:

-- Quarterly status report to the CALS Steering Group
-- Individual issue papers when evaluation of a specific technical issue is completed (incrementally delivered within eight months after go-ahead)

4. Evaluate Air Force experience in applying IDEFO/IDEF1 functional analysis and data modeling techniques, and assess the feasibility of applying this approach on a broad scale for CALS applications. Compare with alternative approaches.

Deliverable:

-- Issue paper (six months after go-ahead)

As DoD needs are determined, via the initial task, adjustments may have to be made to the remaining tasks. Tasks identified by an asterisk (*) appear to be low priority for FY 86. These tasks will be accomplished in FY 86 if possible. If not, they will be deferred to FY 87.

Tentative FY 87/88 Tasks

FY 87 and 88 tasks will be firmed up in the tactical plan delivered six months after FY 86 go-ahead. Tentative tasks include:

FY 87

1. Develop preliminary functional specifications for needed programming language "bindings" to NDL and/or SQL as identified by Milestone 1.

2. Design validation methodology for NDL and/or SQL as identified by Milestone 1.

3. Complete one programming language "binding" to NDL and/or SQL.

4. Demonstrate preliminary data dictionary validation suite on data dictionary shell.

5. Prepare for CALS review a preliminary specification of data dictionary extensions to support:
   graphics;
   distributed databases;
   data modeling.

6. Provide data dictionary shell for use in CALS database design.

FY 88

7. Complete and demonstrate data dictionary validation suite.

8. Complete specifications for data dictionary extensions and begin FIPS processing.

9. Complete and demonstrate NDL and/or SQL validation suite.

10. Submit recommendations to CALS on the applicability of Remote Database Access Service and Protocol to CALS requirements for distributed database.

11. Demonstrate support of graphics by interim data dictionary.

I. DATABASE MANAGEMENT SUPPORT

This section contains a summary of the work accomplished on Database Management Support, tasks II.1.1, II.1.2, II.1.3, and II.1.4 for the FY 1986 NBS statement of work.

A. DATABASE MANAGEMENT SOFTWARE AND STANDARDS

1. ASSESS DOD NEEDS FOR DATABASE MANAGEMENT STANDARDS

In the task to assess the DoD needs for database management standards, there are five specific subtasks. These subtasks are: (1) identify CALS database characteristics, (2) recommend standards, methodologies, and tools, (3) assess standards implementation capabilities, (4) recommend data dictionary strategy, and (5) plan for standards implementation.

The Preliminary Report on Data Management Standards, dated June 20, 1986, provided as a separate report to last quarters report, discussed this task. One of the major shortcomings of this report was that it did not address the specific CALS requirements. At that time, there had been insufficient information available to analyze the specific standards required for CALS. Because of the change in the focus of the CALS effort for the remainder of FY 1986, the tasking in the SOW did not accurately reflect the work that was done or the scheduled deliverables. Consequently, in FY 1987 and FY 1988, the assessment of DOD needs will be done as part of CALS Core Specifications. The contents of the Preliminary Report on Data Management Standards will be a part of the deliverable associated with the development of the CALS Core Specification. Below is a more detailed discussion of each subtask.

a. Identify CALS database characteristics (Subtask II.1.1.a)

The original intent of this subtask was to identify the database characteristics for each of the programs under the CALS umbrella. There are 82 programs described in the CALS draft plans prepared by each of the DoD services. NBS started to document all of these programs and included documentation in the Preliminary Report on Data Management Standards; however, it proved to be impractical and unnecessary for NBS to review all of the programs. Database characteristics and data management requirements can be determined from specific application areas discussed in the NBS Point Paper, CALS Representative Systems. By concentrating on application areas, NBS can determine the commonality of systems and focus on the standards that are appropriate for CALS. This direction will be continued in the FY 1987/88 SOW. The work for this task will be a function of the "Develop CALS Core Specifications" task in NBSs future effort.

b. Recommend standards, methodologies and tools (Subtask II.1.1.b)

The Preliminary Report on Data Management Standards described the data management standards that are available for use in CALS. The June 86 CALS Workshop Report identifies a preliminary finding of the standards required for two application areas, Engineering Data Repository and Printing and Publishing. In the workshop, participants identified a requirement for a Technical Data/Configuration Management capability and a method for indexing technical data which would aid users access to data in these and other applications. Standards such as SQL or IRDS are standards that should be evaluated for potential use in satisfying this requirement. Using the NBS Point Paper, CALS Representative Systems, and the CALS Framework Report (under contract for Feb 87 deliverable) as a basis for identifying CALS standards,. NBS will identify the standards applicable for each application area.

c. Assess standards implementation (Subtask II.1.1.c)

The Preliminary Report on Data Management Standards, Section 4.2, Research and Development, discusses some near term issues and some of the research and development issues that NBS and DOD must address. These will be explored in FY 1987.

d. Recommend a strategy for use of a data dictionary in CALS planning. (Subtask II.1.1.d)

Some preliminary discussion on these topics is in the Preliminary Report on Data Management Standards, Section 5, Strategy for Use of a Data Dictionary. This subtask must be coordinated with the efforts associated with the contract for the CALS Framework and incorporated into the CALS Framework Report.

e. Plans for standards implementation (Subtask II.1.1.e)

An outline was submitted in the last quarterly report and the FY 87 SOW addresses this subtask.

2. ACCELERATE DATABASE MANAGEMENT STANDARDS EFFORT

The goal of this task (II.1.2) is to develop a plan and methodology for accelerating data dictionary and database standards development and validation efforts where they are needed to meet CALS scheduled objectives. Since this task is dependent upon the subtask to assess CALS needs for standards (subtask II.1.1.c) which is not yet complete, the work on this task will be deferred until FY 87.

One significant event that occurred during this quarter was the vote by OSI TC97/SC21/WG3 IRDS Rapporteur Group to register the ANSI X3H4 IRDS Command Language and Panel Interface Document as a Draft Proposal (DP) International Standard (IS). There had been strong opposition to the U.S. position on the proposal; however, through outstanding efforts and persuasion by the U.S. delegates it was finally approved as a DP IS. If this document had not been approved, there would have been an estimated 9-18 month delay in progressing the IRDS through the international standards process. To support the joint DOD/NBS CALS effort, NBS issued a contract to Dr. Henry C. Lefkovits, the principal contributor to the IRDS Specification, to assist NBS and X3H4 in accelerating progression of the IRDS to a DP IS.

3. EVALUATE DOD DEMONSTRATION PROGRAMS

The goal of this task (II.1.3) is to evaluate DOD demonstration programs in support of tasks II.1.1 and II.1.2. Because these two prior tasks will be part of the FY 1987 effort, this task will be accomplished in conjunction with other FY 1987/88 tasks.

4. EVALUATE IDEF0/IDEF1

The goal of this task (II.1.4) is to evaluate Air Force experience in applying IDEF0/IDEF1 functional analysis and data modeling techniques, and assess the feasibility of applying this approach on a broad scale for CALS applications. Because of the change in the focus of CALS for the future, nothing was done on this task and it is being dropped from the planned future work statement.

COMPUTER AIDED LOGISTICS SUPPORT (CALS)

DATA MANAGEMENT STANDARDS
PRELIMINARY REPORT

Prepared For

Working Group on Specifications and Standards

Prepared By

Patricia Konig
Frank Spielman
Joan Sullivan

National Bureau of Standards
Institute for Computer Sciences and Technology

June 20, 1986

TABLE OF CONTENTS

COMPUTER AIDED LOGISTICS SUPPORT (CALS)
DATA MANAGEMENT STANDARDS
PRELIMINARY REPORT

1. INTRODUCTION

This preliminary report is an interim deliverable for task II, Database Management Support, of the National Bureau of Standards (NBS) proposal. The report discusses four specific tasks: 1) database characteristics, 2) data management standards, methodologies, and tools, 3) data management issues, and 4) strategy for use of a data dictionary in CALS.

For the short period of time that NBS has been working on the CALS effort, there has not been sufficient time to adequately analyze the requirements in enough depth to recommend the standards needed to support CALS. Consequently, the emphasis on this preliminary report is on the existing data management standards and where they can be use in the existing CALS applications. Future activity and reports will focus on the standards needed to support the common DOD-wide requirements for CALS. NBS will then be able to determine the appropriateness of the existing standards, the standards which need to be enhanced, and/or any new or DOD tailored standards required for CALS.

2. DATABASE CHARACTERISTICS

There are two major methods of grouping the CALS programs/systems described in the Service Implementation Plans. First, there are the technical data representations and secondly there is the grouping by functional area. Each of these views is described below.

2.1. TECHNOLOGY VIEWS

Future CALS systems focus on the four basic automation capabilities depicted in Figure 1. This figure and supporting material is based on information obtained from the "U.S. Air Force Plan for Implementation of CALS." Each of these areas has tended to develop as a separate island of technology, not well connected to the others. There is a very large infrastructure of large-scale, batch processing and communication oriented systems together with the related human investments and administrative and technical procedures necessary to operate them. The scale and complexity of the systems, the resources necessary to make significant change, and the rapid change of the underlying technology all represent major challenges. Accordingly, special emphasis must be given in the CALS program to the technology and standards activities so that CALS systems can evolve as the technology changes. Existing standards and protocols must be

enhanced or new ones developed to support the interchange of data within and between these four islands of automation.

The four technical views depicted in figure 1 are: Image, Text, Alphanumeric, and Communications. The communications view is being addressed in other CALS initiatives and thus is not a part of this NBS/DoD effort. The image view is really a graphics representation of a part or component of a weapon system. The text view is considered to be long textual material such as a report or manual. The alphanumeric view is highly structured data that would be found in an inventory database.

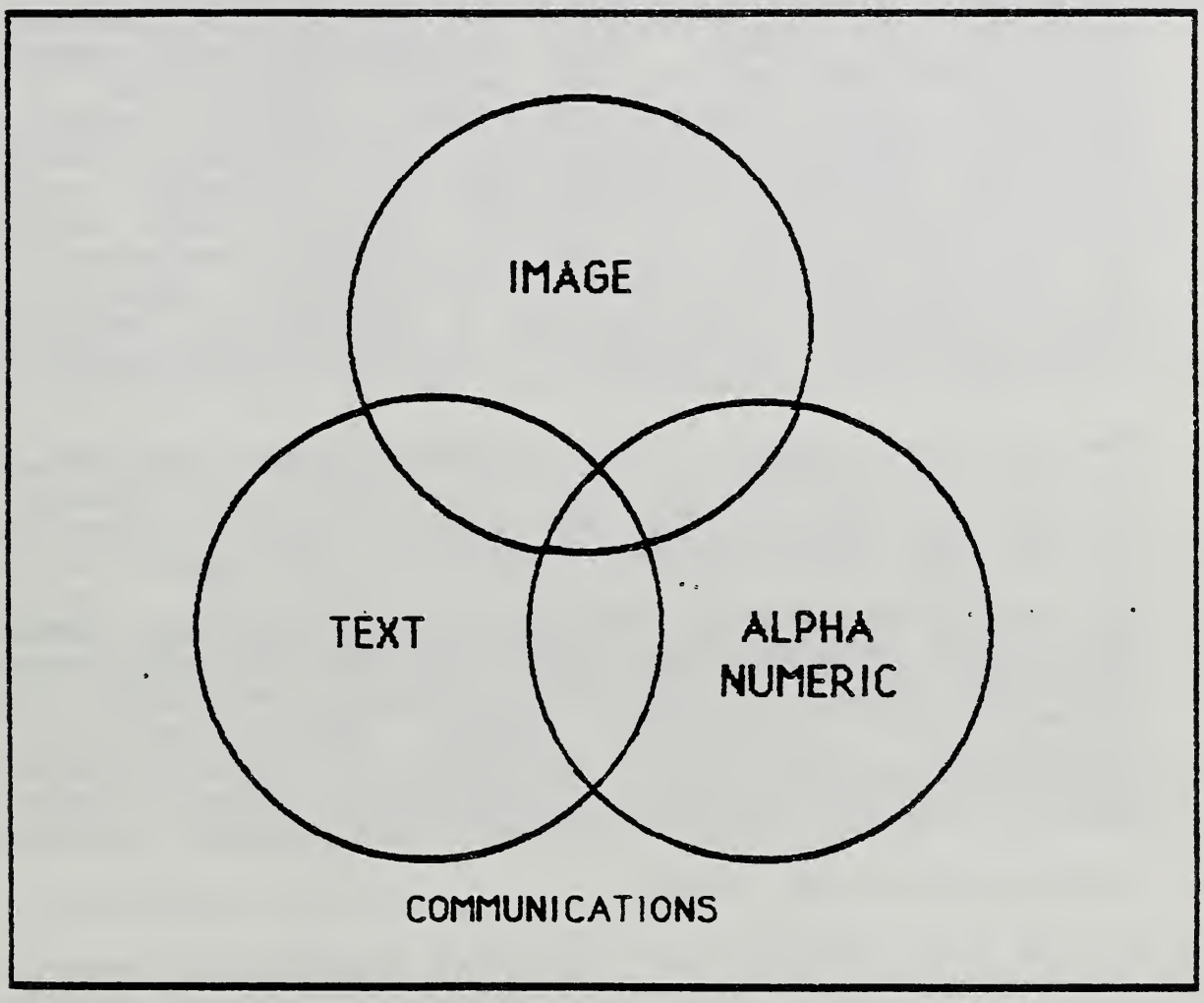


FIGURE 1 BASIC CALS TECHNOLOGIES

2.2. FUNCTIONAL VIEWS

The functional area or view groups the programs into three basic categories: Product Data, Technical Support Data, and Logistic Support Data. Each of these functional views is described in more detail below.

The Product Data is the totality of data elements which completely define the product for all applications over its expected life cycle. This includes the data about the engineering drawings and the semantic description of these drawings, such as tolerance levels, material composition, geometry, topology, attributes, and features necessary to completely define a component part or an assembly of parts. This data is image or graphics oriented with supporting semantic descriptions and is increasingly developed through the use of CAD/CAE systems. The semantic description is often referred to as text but in reality this data may be more structured and therefore appropriately stored in a structured file or database.

The Technical Support Data consists of data that is used to develop and deliver technical and operational manuals. It is largely textual (free form) data, however it usually will include some drawings for illustration purposes.

The Logistics Support Data consists of the data needed to supply and maintain the weapon systems in an operational readiness state. Logistics support generally include elements such as training, supply, support equipment, and maintenance. This data is primarily structured and is sometimes referred to as alphanumeric data. There are several existing automated systems that support this area.

2.3. DISCUSSION OF THE VIEWS

Some of the CALS programs may primarily support only one of the major functional areas but still require the use of several or all of the technical views. The relationship between these two views is described below.

For the Product Definition Data, there are essentially two areas that are being automated: Engineering Repositories and CAD/CAM data. The engineering repositories are in the process of automating the storage and retrieval of the engineering drawings. Although the drawings are or may be the result of CAD/CAM systems, there is a requirement to use a structured database for maintaining an on-line index to the drawings. For example, the Army and Air Force DSREDS/EDCARS system is using IBM's IMS Database Management System (DBMS) to maintain and retrieve index data about the drawings. Section 4 discusses the future use of standard database and data dictionary systems in supporting CALS.

For the Technical Support Data, there are also two areas where standards may be used to enhance projects such as ATOS, 600S, and NAPPS. Although the data is largely textual, there appears to be a need to include certain logistics support data related to component parts as well as index data for the large textual files. Also, there appears to be a requirement to interface these automated systems with other logistics support systems. The use of database and/or data dictionary standards can help facilitate this interface.

The Logistics Support Data is a prime candidate for evolving database and data dictionary standards to support projects such as UDB. Since the data is highly structured, these standards should be evaluated for applicability to enhance the access to and interchange of logistics data. A standard data dictionary would greatly facilitate a user in locating logistics data in a distributed environment. It also would provide assistance in locating parts when specific information such as part number, nomenclature, etc., is not known.

2.4. PHYSICAL CHARACTERISTICS

This section will include a description of the physical characteristics of the data required for each of the CALS programs. NBS hopes to obtain more information about the physical characteristics through additional visits to CALS related installations and/or CALS workshops. Some of the areas to be covered in this section of the final report include:

- Types of data -- textual, image, structured data (i.e., stock number, indexes); identify type of data required for input, storage, and output
- Distributed or centralized data
- Update or query driven
- On-line versus batch updates and queries
- Indexing and cross referencing requirements for a specific system and for interface support to other systems
- Volume of storage required
- Interfaces to other systems for data sharing or interchange
- Use of, or plans to use, standards (local, service, ANSI, ISO, etc.) for data dictionaries, database management systems or data interchange

- Name of any current data dictionary and/or database management systems
- Security requirements
- Special disaster recovery requirements

3. DATA MANAGEMENT STANDARDS, METHODOLOGIES, AND TOOLS

There are several standards available or being developed which can support the management of CALS data. In addition, there are different methodologies and tools that should be evaluated for use by the CALS programs. The data management standards, methodologies, and tools are described in the following paragraphs.

3.1. DATA MANAGEMENT STANDARDS

Rapid increases in the costs associated with software development and maintenance are driving organizations to alternative methods of data management and applications development, including commercially available software, DBMSs, and application generators. Consequently, the advantages of standards for software facilities and interfaces are beginning to be recognized, in much the same manner as in the computer communications field.

Users of sophisticated data management software need to export their data to powerful graphics and other software systems. Organizations who employ independent data dictionary systems need to control data used by a DBMS, computer programs, and so-called "fourth generation languages." Those who purchase data management technology expect these expensive software facilities to support constantly changing requirements. Just as for hardware systems, the days of user dependence on one vendor for all of their software needs are technically and economically impractical. The overall cost of managing data can be decreased through the use of data management standards by: (1) aiding the transport of personnel skills between organizations and information systems therefore increasing productivity and (2) more effectively interchanging data between systems.

CALS' objectives include access to data at various nodes of a distributed, heterogeneous database. The users on a network should be required to know at most two database languages (a local and a global language). Most users should be able to work with at most two simple schemas (local and part of the global schema); users requiring more flexibility should be able to explore the global schema, subject to security constraints, using menus or other aids.

Data management standards provide a foundation for achieving the objectives outlined above. There are existing standards related to data interchange; but at present, there are no existing international, national, or Federal database management software standards. However, proposals are currently under review within the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO) technical committees. These proposals address four critical areas for standards: (1) data structures and languages; (2) dictionaries for managing and controlling complex data descriptions; (3) data interchange; and (4) distributed data environment.

3.1.1. DATABASE STRUCTURES AND LANGUAGES STANDARDS

Over the last few years, considerable research and development has been done on different database models including network, hierarchical, inverted files, and relational models. Proposals have been developed to specify database languages for the network and relational data models. The background, benefits, environment and standards relating to database management systems are discussed in this section.

3.1.1.1. DBMS BACKGROUND

The late 1960's and 1970's brought a flurry of database research. Charles W. Bachman is widely recognized as one of the early developers of the network approach to data management. This approach provides a flexible scheme for linking together records of different types using a pointer-chain structure. In 1971, the Data Base Task Group of the CODASYL COBOL Committee proposed network database languages for schema and subschema data descriptions as well as for data manipulation. The Accredited Standards Committee X3H2 adopted these proposals in 1978 as a basis for development of a standard language for network structured databases.

Hierarchical systems evolved independently in the 1960's so that currently there are many different versions in the marketplace. Since the hierarchical model is a special case of the network model, there is no separate standardization effort for a hierarchical database language.

End-user oriented DBMS's, characterized by an inverted file access technique, proliferated in the 1970's. However, due to inconsistent data structures and lack of a common data manipulation strategy, there has been no attempt at standardization.

In 1970, E. F. Codd wrote the seminal paper that defined the concepts of normalization and joins for relational tables. This paper created a lot of interest in various high level query and manipulation languages based on predicate calculus. Structured

Query Language (SQL) using the relational model was proposed in 1974. Over the next five years, IBM built and evaluated a prototype implementation called System R. The technology developed in that prototype was implemented in 1980 as SQL/DS and then in 1983 as DB2. The SQL database approach was widely emulated by hardware and software vendors. In 1982, the scope of work for X3H2 was expanded to include a standard for relational database management systems.

3.1.1.2. DBMS BENEFITS

The purpose of database language standards is to promote portability of database definitions and database application programs between different installations. Additional objectives are:

- to encourage more effective utilization and management of data by users (both end-users and programmers) by ensuring that skills acquired on one job are transportable to other jobs,

- to ensure that there will be a pool of trained personnel available to work on new development projects as well as to maintain existing systems,

- to reduce the cost of software development by achieving increased programmer productivity through the use of database technology employing standard structures and operations, standard data types, standard constraints, and standard interfaces to programming languages,

- to protect the software assets of the organizations by ensuring to the maximum extent possible that database management systems standards are technically sound and that subsequent revisions are compatible with the installed base.

3.1.1.3. DBMS ENVIRONMENT

The DBMS approach to computer systems is an organized effort to share structured data across a variety of applications. The data typically consists of a large number of named data elements (such as numbers and short textual fields). Data elements are grouped into various record types, with related data elements being grouped together. Many interactive users, as well as batch programs, access the data and modify the data concurrently. As soon as one authorized user changes the values in the database, all the authorized users can view the new data. Redundant data is kept to a minimum. Redundant data often results in out-of-date and inconsistent data for applications which use the data but are not responsible for creating it.

Of course, the DBMS software has security features to prevent unauthorized individuals from viewing or updating the data. The DBMS also has strategies for allowing many users to update data concurrently without causing confusion and corrupting the database. The DBMS enforces integrity constraints and will not allow certain invalid data values to be entered into the database. The DBMS has utilities to restore the data (almost completely to its original state) when the computer goes down in the middle of heavy updating or when a disk is damaged and becomes un-readable.

In addition, the DBMS is a productivity tool. The DBMS allows the end-user to obtain reports without writing a program. The DBMS provides an interface to programming languages such as COBOL. This allows the programmer to simplify his program by factoring out most of the logic which deals with data access. This logic, coded in a procedural style, is replaced with requests for database services which are coded in a more efficient, less error-prone, non-procedural style. In fact, the DBMS is the major component in an integrated set of productivity tools managed by a dictionary. The dictionary (the subject of a separate standard) is an essential part of the DBMS, yet it performs many other functions as well.

Each DBMS vendor offers a different mix of productivity tools, and perhaps a different database language. When a user needs to share data or programs with other users or when he has to convert his system from one vendor's computer to another vendor's computer, the greatest expenses are incurred in converting the data, converting the application programs (written in COBOL, FORTRAN, etc.), and retraining personnel. Current standards activities for dictionary and database languages focus on the last two concerns.

There are two standards for database languages which assume different underlying data models and are best suited for different types of applications. In many cases, an application could be created using either model. The two standards are the Network Data Language (NDL) and the Structured Query Language (SQL), respectively. The database language specifications for NDL and SQL are expected to become ANSI standards X3.133-1986 and X3.135-1986 respectively. The final ballot closes on June 16, 1986. Both NDL and SQL are being processed by ISO as Draft International Standards (DIS) and by NBS as Federal Information Processing Standards (FIPS).

3.1.1.4. NDL APPLICABILITY

The network model is appropriate for highly structured applications requiring rapid access along predefined paths. The network model is desirable for transaction-oriented processing, where a small portion of a very large database is accessed in

real time, and data from several related records must be combined to perform the function.

An example of access along a predefined path would be the function of locating a book in a library. From a subject descriptor (e.g. "optical disk"), several titles are located. For each acceptable title, the call number of the book is located. For each call number, the locations of copies of the book are determined. If the book is available now, a check-out procedure is followed. If not, perhaps a reservation procedure is followed. This sequence of events happens over and over, and it would be important in a large on-line system that the data access be as efficient as possible.

3.1.1.5. NDL SPECIFICATION

The NDL is a specification for the logical data structures and basic operations of the network model. The network data model contains two basic data structures: the record and the set. The record is the basic unit of data manipulation. Records are stored, erased, found, modified, and connected to and disconnected from other records. The set is the basic unit of navigation. A user is able to move directly from one record to another along logical set access paths defined by the database schema. The navigation path has to be determined during the design of the database. Because of this, ad hoc queries (requiring data from records which are not connected by a set relationship) may be very inefficient.

The NDL is a specification for interfacing DBMS software to standard programming languages. The NDL is a programming language intended for use by applications programmers who have been trained to use the network model. Despite its complexity, the network model has payoffs in the areas of performance and enforcement of inter-record integrity constraints. A network model DBMS will probably have a user-friendly query langauge, but it will not be in the style of NDL.

The NDL standard has two levels of conformance, as well as the possibility of functional conformance (with vendor-supplied syntax). NBS recommends, in the Federal Information Processing Standard (FIPS) for NDL, that only Level 2 (the full standard) be used. The NDL standard specifies interfaces to four standard programming languages -- COBOL, FORTRAN, Pascal, and PL/I.

3.1.1.6. SQL APPLICABILITY

The relational model is appropriate for applications requiring flexibility in the data structures and access paths of the database. The relational data model is desirable where there is a substantial need for ad hoc data manipulation by end-users who are not computer professionals, in addition to the need for

access by large operational systems. Prototyping and decision support systems can take advantage of the flexibility of the relational model. However, performance is still a problem, although recent releases of several products offer substantial gains in performance. The use of a database machine especially designed for database operations may solve performance problems.

3.1.1.7. SQL SPECIFICATION

The SQL is a specification for the logical data structures and basic operations of a relational data model. The primary data structure of the relational model is a table. A table is defined in terms of rows and columns. A row is a non-empty sequence of values (a record). A column is an unordered collection of values (all the values of a given data element). There is no predefined navigation that the user has to follow to find the desired data. The user can obtain data from any number of tables by joining them together at the time he is retrieving the data. Thus, the relational model is considered to be more dynamic and flexible.

The SQL standard also specifies interfaces between the DBMS software and standard programming languages. SQL is programmer-friendly, since it provides a high-level, nonprocedural way to create, access and modify data from within a standard programming language. In addition, the same language can be executed interactively to obtain ad hoc reports. Because end-users are comfortable thinking of their data as rows and columns in tables, and because the data manipulation language of SQL is English-like, SQL can be used successfully by non-programmers without extensive training.

The SQL standard has two levels of conformance; the FIPS recommendation is again the Level 2 (full standard). SQL specifies interfaces to four programming languages -- COBOL, FORTRAN, Pascal, and PL/I. In addition, conformance to SQL may be claimed for implementations which process data manipulation statements directly.

X3H2 is continuing the development of additional SQL features. Among the expected features are interfaces to programming languages Ada and C, referential integrity (inter-record integrity constraints), date/time data types, browsing capabilities, etc.

3.1.1.8. NDL/SQL USAGE

The structures and operations specified by NDL and SQL are typical of existing capabilities in a wide variety of database management system products and are expected to be fully supported by vendors in off-the-shelf products by April 1988 (the end of the transition to the FIPS). These specifications should be used as a starting point in the design and implementation of

appropriate (or applicable) CALS systems. They should also be considered during development of standard interfaces to other standard products such as data dictionary, graphics, etc.

3.1.2. STANDARD FOR MANAGING DATA DESCRIPTIONS

In recent years, the concept of Information Resources Management (IRM) has gained popularity. This philosophy recognizes that information and all aspects of its production, dissemination, control and cost should be treated as an enterprise resource, just like money, real property, and people. Thus, the need to manage and control the totality of the enterprise's information resources was recognized. In order to support this need, the concept of an Information Resource Dictionary System (IRDS) has evolved. The IRDS will serve as a tool to help manage and model the enterprise's information environment. Managing this environment requires the capability to document an enterprise's information environment, to maintain an inventory of all information entities, to support the operational aspects of the information environment, to illustrate the interrelationship of information entities, and to document the location of the entities in the information environment.

The draft IRDS Standard contains the specifications for a software package that can be used to manage an enterprise's information environment. The IRDS Specifications are a draft proposed American National Standard (dpANS), a draft proposed U.S. Federal Information Processing Standard (FIPS), and a Working Document of the International Organization for Standardization (ISO). NBSIR 85-3164, A Technical Overview of the Information Resource Dictionary System, provides a technical overview of the computer software specifications for an Information Resource Dictionary System (IRDS). It summarizes the data architecture and the software functions and processes of the IRDS.

3.1.2.1. IRDS STANDARD BACKGROUND

In 1980, both ANSI and NBS of the United States Department of Commerce initiated efforts to develop standards for dictionary software. The ANSI effort began with the approval by the American National Standards Committee for Information Systems (X3) of a project to develop a standard for an "Information Resource Dictionary System" (IRDS). This resulted in the July 1980 convening of Technical Committee X3H4 responsible for developing the standard for an IRDS.

In 1980, the NBS initiated an effort to develop a Federal Information Processing Standard (FIPS) for Data Dictionary Systems. Since that time, NBS has actively pursued a standard IRDS through a series of events which are summarized as follows:

Prepared and disseminated the Prospectus for Data Dictionary System Standard in 1980 which discussed the use of data dictionary systems and described plans for developing a FIPS.

Conducted four Data Base Directions workshops that investigated Information Resource Management, Database, and Data Dictionary related issues.

Conducted interviews with U.S. Federal agency personnel to identify requirements for a Data Dictionary standard and published the Federal Requirements for a Federal Information Processing Standard Data Dictionary System.

Conducted a series of six workshops for representatives of more that fifty U.S. Federal agencies to obtain feedback on evolving Data Dictionary Specifications.

Prepared and disseminated an interim publication, Functional Specifications for A Federal Information Processing Standard Data Dictionary System, for review and comment early in 1983; NBS received and analyzed comments from over 100 U.S. Federal Government agencies.

Prepared and disseminated the draft Specifications for the proposed Federal Information Processing Standard for Data Dictionary Systems in August 1983.

Sponsored two workshops for vendors to review the Specifications as they evolved. Vendors representing seventeen of Data Dictionary Systems participated in the workshops.

Disseminated the interim publications and draft Specifications to more than 200 private industry organizations, universities, and state and local governments in the U.S. and organizations in foreign countries.

Although ANSI X3H4 and NBS used different titles (i.e., "Information Resource Dictionary Systems" and "Federal Information Processing Standard for Data Dictionary Systems"), the two groups had identical goals and a similar development approach.

The two efforts came together in September 1983 when X3H4 voted to adopt the August 1983 version of the draft Federal Information Processing Standard for Data Dictionary Systems as its Base Document. Since that time, the Base Document has evolved to its present form as a draft proposed American National Standard (dpANS) and a draft proposed FIPS for an IRDS.

The dpANS and the FIPS IRDS underwent public review and Federal agency review in the latter part of 1985. Accredited Standards Committee X3H4 made changes to the IRDS specifications, based on the public and Federal review comments, during the first four months of 1986. Another public review period, of the recent changes, will take place June 2 - September 2, 1986. NBS expects that the IRDS, number X3.138, will become an American National Standard and a FIPS in early 1987.

In January 1984, the ISO Technical Committee 97 approved Work Item 97.21.6 for IRDS. The Work Item is assigned to Subcommittee 21, Working Group 3 (TC97/SC21/WG3) for the purposes of progressing it to become a future International Standard.

As a result of the joint DoD/NBS CALS effort, NBS will issue a contract to Dr. Henry C. Lefkovits, the principal contributor to the IRDS Specification, to assist NBS and X3H4 in formulating a U.S. position that will help accelerate progression of the IRDS to a Draft Proposed (DP) International Standard. The recommendation to progress the IRDS to DP status will be voted upon at the September 15-19, 1986 meeting of TC97/SC21/WG3.

3.1.2.2. PROPOSED IRDS STANDARD

The IRDS Specifications contain the most commonly used facilities of existing data dictionary systems and represent a state-of-the-art level of technology in dictionary systems. The proposed base level IRDS Standard includes specifications for a "Core" dictionary system plus specifications for optional "Modules". Although the IRDS Modules interface with the Core, they are independent of one another. Organizations, therefore, need to acquire only those modules that support their requirements. After this base level standard is finalized, development of additional modules will begin.

The core IRDS contains the basic capabilities that organizations generally need. These core specifications are intended for implementation on large microprocessors and small minicomputers as well as large computers. The five optional modules in the IRDS, that provide additional facilities, contain specifications for: Basic Functional Schema; IRDS Security; Extensible Life Cycle Phase; Procedure Facility; and Application Program Interface.

To provide additional flexibility in the use of the core IRDS, the specifications allow for the capability to customize or extend the types of data that can be stored in the dictionary. Organizations can use this capability to describe unique resources and define organization specific system development methodologies.

The core IRDS contains two user interfaces: a menu-driven "Panel" interface and a "Command Language" interface. The Panel interface is designed to support interactive processing, especially by inexperienced dictionary system users. This interface leads users down a structured path of screens (panels) that result in the execution of dictionary system functions. Thus, non-technical staff as well as technical staff will be able to execute certain core IRDS functions without having to understand or use the more complex syntax of the Command Language interface. The Command Language interface may be used in either a batch or interactive mode. Because each of the user interfaces will be similar in all IRDS implementations, individuals will be able to use different IRDS systems without extensive retraining. A vendor is in compliance with the core standard and three of the five modules if only one of the user interfaces is provided. Two of the modules, the Procedure Facility and the Application Program Interface require the Command Language. If both user interfaces are provided, both must comply with the standard.

The IRDS Specifications include an IRD to IRD Interface Facility which provides a controlled method of moving data from one standard dictionary to another standard dictionary. Organizations using the standard IRDS could, for example, extract data from decentralized dictionaries and add it to a central dictionary that focused on organization-wide data management. The specified IRD to IRD Interface supports this transportability of data even in the case where the standard IRD systems are developed by different vendors and are resident on different hardware systems at different locations.

3.1.3. DATA INTERCHANGE

Data interchange standards have been developed to provide a mechanism for data structures, structured database and files, to be easily moved from one computer system to another, independent of vendor or model. Data structures to be interchanged can vary significantly in complexity and size. There is a need for a common method to accomplish these interchanges. It is also desirable that any medium (such as a communication line, a magnetic tape, a disk pack, a flexible disk, etc.) can be used for the physical interchange. If possible, all information necessary to successfully recreate the structure in the target system should be contained within the information transported.

To meet these needs, two standards have been developed which are: 1) Specification for a Data Descriptive File (DDF) for Information Interchange, ANSI/ISO 8211, and 2) Abstract Syntax Notation.1, ISO Draft International Standard (DIS) 8824/8825.

3.1.3.1. DATA DESCRIPTIVE FILE (DDF)

The DDF standard establishes media and machine independent formats and data record formats for interchanging information between computing systems. The standard is intended for use with physical recorded media as well as with communications media. The contents in the user data structure can be of any internationally recognized character set and coding and are interchanged in a transparent fashion. The intermediate structure through which the information passes is designed for interchange purposes only and is not intended to be used for general processing.

The approach used for the standard was to define an interchange format into which the sender's information is mapped and conveyed to the receiver's system. Upon receipt of the information in the interchange format it is then mapped into the receiver's format without loss of structure and content. The standard specifies a method to describe a robust interchange structure which can accept most user data structures.

Most data structures in common use can be described and interchanged using this standard. The structures within the interchange file can be of the following forms: elementary data, vectors, arrays, and hierarchies. The elementary data may be character strings, bit strings, or various numeric forms. User file structures such as sequential, hierarchical, relational and indexed can be converted into the interchange structure. Network structures can be interchanged but additional pre-processing and post-processing is necessary to preserve logical linkages.

The standard provides for three interchange levels from which the users may choose, based on the complexity of their data structures. The first interchange level supports multiple fields containing simple, unstructured character strings. The second level supports fields containing structured data and a variety of data types. The third level supports hierarchical data structures.

3.1.3.2. ABSTRACT SYNTAX NOTATION.ONE (ASN.1)

ASN.1 is an ISO Draft International Standard (DIS), ISO DIS 8824/8825. It is more general than the DDF interchange forms in that it is a "language" for defining general data structures rather than a syntax for specifying columns and row occurrences of an underlying tabular structure. ASN.1 allows for the interchange of any data structure that the user specifies. The specification for the language syntax is separate from the basic encoding rules. The language syntax, ISO DIS 8824, provides the specifications that a user must use to define the data structure. The basic encoding rules, ISO DIS 8825, specify the encoding methodology for the data contained in the data structure. Using

this approach, information can be exchanged in two parts. The first part is a character string that defines a specific data structure, and the second part defines a string of octal numbers that is an encoding of the values in the defined data structure.

The interchange of any data structure defined by the ASN.1 syntax is accomplished using three elements: Type, Length, and Value. The Type is a bit sequence that identifies a data type previously defined by the language syntax. The Length is an integer that declares the length in bytes of a data occurrence. The Value is the actual encoding of the data occurrence. Any data structure definable via the ANS.1 syntax can then be encoded as a nested hierarchy of type/length/value triples.

The ISO ASN.1 DIS uses the CCITT X.409 Message Handling Facility notation. However, the ASN.1 provides additional alternatives to the language and syntax notation that are not available in the X.409 standard.

3.2. METHODOLOGIES

CALS will require the use of several different methodologies. There must be methodologies to support the way that the information systems are to be constructed and later maintained. The life cycles of the information systems must be established and integrated with weapon system life cycle management. The configuration of information systems also must be managed. The data dictionary , ultimately the standard IRDS, is the tool that will facilitate the management of these methodologies.

3.2.1. DATA MODELING

Data Modeling is the process of constructing a logical representation of data and the associations among data. The data model can be used to represent data structures throughout an organization, or can represent a single logical database structure. It describes the data structure independent of the targeted hardware or software environment. The data model of an organization is frequently referred to as the organization's information model or business systems model. A person or group of people responsible for constructing a data model must analyze the information requirements, develop diagrams and charts, and coordinate systems development with the end-users, database administration, and data processing staff.

There are numerous data modeling methodologies that are available. Computer support for data modeling is almost mandatory, given the size and complexity of the data model and the various external views, as well as the frequent changes resulting from the new requirements which are folded into the model on a continuing basis. There is a real need for a consistent approach and notation, as well as standard automated

support tools and interchange formats. Standards for data modeling should include data modeling methodology, IGES, SGML, IRDS, SQL and possibly NDL. NBS has written a guide on logical database design (data modeling) which describes a generic methodology for building a data model. The Guide and more specific data modeling techniques are discussed below.

3.2.1.1. GUIDE ON LOGICAL DATABASE DESIGN

NBS Special Publication 500-122, Guide on Logical Database Design, discusses an iterative methodology for Logical Database Design. The guide describes the design methodology for determining the hardware-independent, software-independent structure for an organization's data requirements. The objective is to decrease the effort required to maintain organizations' information processing systems.

3.2.1.1.1. BACKGROUND:

Logical Database Design (LDD) is the process of determining the fundamental data structure needed to manage an organization's information resource. LDD provides a structure that determines the way that data is collected, stored, and protected from undesired access and modification. Since data collection, storage, and protection are costly, and since restructuring data generally requires expensive revisions to programs, it is important that the LDD be of high quality.

A high quality LDD will be: (1) internally consistent, to reduce the chances of contradictory results from different information systems; (2) complete, to ensure that known information requirements can be satisfied and known constraints can be enforced; and (3) robust, to allow adaptation of data structures in response to foreseeable changes in the information requirements. To fulfill these considerations, a LDD should be: (1) independent of any particular application, so that all applications can be satisfied; and (2) independent of any particular hardware or software environment, so that the data structure can be supported in any environment. A good LDD will ensure that modularity, efficiency, consistency, and integrity are supported in the subsequent operational database.

3.2.1.1.2. LDD METHODOLOGY:

The Logical Database Design (LDD) methodology described in the Guide includes four phases: Local Information-flow Modeling, Global Information-flow Modeling, Conceptual Schema Design, and External Schema Modeling. These phases are intended: (1) to make maximum use of available information and user expertise, including the use of a previous Needs Analysis and (2) to prepare a firm foundation for physical database design and system implementation. The methodology recommends analysis from

different points of view--organization, function, and event--in order to ensure that the logical database design accurately reflects the requirements of the entire population of future users. The methodology also recommends use of a data dictionary system, preferably a standard IRDS, in order to conveniently and accurately handle the volume and complexity of analysis and design documentation. The report places the methodology in the context of the complete system life cycle.

3.2.1.2. DATA MODELING TECHNIQUES

CALS is currently using data modeling methodologies to support the CAD/CAM area. For example, the Air Force is using IDEF1 in the IDS program. IDEF1 is a top-down approach using the Entity Relationship technique. The NBS AMRF is using another technique called SAM*. NIAM is another technique which is a bottom-up approach to data modeling. A description and discussion of these methodologies will be included in the final report.

3.3. TOOLS

In the arena of data management tools, there are several commercially available tools that can be used by CALS. These tools can be grouped into Database Management Systems (DBMS), Data Dictionary Systems, and Data Modeling. In the discussion on these tools, some vendors and commercial products are listed. The inclusion or omission of a particular company or product does not imply either endorsement or criticism by the NBS.

NBS has worked with most of the vendors cited in the following paragraphs both on the relevant Accredited Standards Committees and by conducting vendor workshops. The forthcoming NDL, SQL, and IRDS standards reflect a technical consensus of the major vendors. None of the vendors have yet made formal announcements that they intend to conform to the standards. However, several vendors have told NBS confidentially that they plan to either change their existing system or develop a new system in order to conform, principally to the SQL and IRDS standards. Some vendors also are developing conversion software that will help their existing customers migrate from the current system to the standard data dictionary/database system. NBS will provide additional information on this subject in the <u>Plan to Expedite the Development and Implementation of Data Management Standards</u>.

There are many commercially available DBMSs on the market. Many of the vendors support the relational model, especially in the microcomputer environment. Most of the vendors of these products will partially support SQL. Most packages will not run on both a micro or mainframe; however, ORACLE is one example of a package that will run on both microcomputers and mainframe computers. Other products which run under UNIX operating systems will run on either mainframe computers or microcomputers operating under a

UNIX system. There are vendors that support the network data model; however these systems primarily run on mainframe computers. Some mainframe DBMS software packages described in the Auerbach Data Base Management publication include:

Vendor	Product
Cincom Systems, Inc	TOTAL
Applied Data Research, Inc	ADR/DATACOM/DB
Cincom Systems, Inc	TIS
IBM Corp	IMS/VS
Computer Corp of America	Model 204
Cincom Systems, Inc	ULTRA
Infodata Systems, Inc	INQUIRE
Burroughs	DMS II
Oracle Corp	ORACLE
IBM Corp	SQL/DS
Relational Technology, Inc	INGRES
Mathematica Products Group	RAMIS II
Software AG	ADABAS
IBM Corp	ADF-II

There are several data dictionary systems on the market. Some come with DBMS packages while others are stand alone or independent systems. Some vendors even offer products that support an integrated environment, where the data dictionary, DBMS, and other functions communicate with each other. Some packages include:

Vendor	Product
Applied Data Research, Inc	Datadictionary
Cincom Systems, Inc	TIS Directory
Computer Associates International, Inc	Integrated Data Dictionary
Computer Corp of America	Dictionary/204
Cullinet Software, Inc	Integrated Data Dictionary
D&B Computing Services	Nomad2 Data Dictionary
Infodata Systems, Inc	Edict
Information Builders, Inc	Focus Data Dictionary
IBM Corp	DB/DC Data Dictionary
Manager Software Products (MSP), Inc	Datamanager
Martin Marietta Data Systems	RAMASTER
M. Bryce & Associates, Inc	Information Resource Manager
Oracle Corp	Integrated Data Dictionary
SAS Institute, Inc	Integrated Data Dictionary
Software A.G. of N.A., Inc	Predict
TSI International	Data Catalogue II
Uccel Corp	UCC Ten Data Dictionary

The above list of vendors was extracted from a report, Data Dictionaries: Knowledge-Bases for the Future, prepared by International Data Corporation (IDC).

Some vendors offer products that support data modeling. The data modeling support ideally should be integrated with a data dictionary system. Some of the vendors that support data modeling with their data dictionary systems include MSP's DESIGNMANAGER and Data Catalogue II's FACET.

4. ASSESSMENT OF STANDARDS FOR CALS

The data management standards must be assessed in terms of current, intermediate and long term capabilities in meeting CALS objectives. Some of the issues involving standards and the CALS effort are addressed below.

4.1. DDF/ASN.1

There are some issues that may impact the near term implementation or selection of standards related to CALS projects. One of the issues that has a near term impact is in the area of data interchange, primarily for structured data. That is, should a DDF or an ASN.1 be used for data interchange? If both data interchange standards are used in CALS, what impact will this have? In the IRDS specification, the data interchange between IRDSs must be done using the DDF standard. However, the interchange of engineering data, as specified by NBS' Automated Manufacturing Research Facility (AMRF) effort, is done using ASN.1.

Further analysis is required to determine if both should be used in the respective areas or whether only one should be used. Assessments on the use of data interchange standards such as DDF or ASN.1 are currently not explicitly included in the NBS/CALS statement of work.

4.2. RESEARCH AND DEVELOPMENT ISSUES

There are several research and development issues that NBS/CALS must address. The issues identified to date are: (1) Distributed databases and IRDS, (2) Integration of Graphics standards with the IRDS standard, and (3) Data Modeling support by the IRDS.

4.2.1. DISTRIBUTED DATABASE ENVIRONMENT

The overall objective of CALS is to be able to access data from various nodes of a distributed, heterogeneous environment. In other words, a user logged onto a system using a remote terminal could first determine what data is available and secondly could

then access the data without being concerned about where that data is located or the characteristics of that data.

First of all, a strategy for the use of a data dictionary or IRDS needs additional evaluation, research, and the subsequent development of another IRDS module. The existing IRDS will provide the CALS users with the necessary knowledge to determine what data is available and where it is located. The strategy for using the IRDS is discussed in the next section. Specifications for supporting database management systems in a distributed environment also need to be developed.

Additional evaluation and research is needed in the area of updating and retrieval of distributed data. Within ISO, the Remote Data Access (RDA) Rapporteur Group of TC97/SC21/WG3, is using, as their base document, the Technical Report, European Computer Manufacturers Association (ECMA) TR/30. This report is a specification for a Remote Data Access Service and Protocol (RDASP). It defines: (1) a database model, (2) operations on the database model, and (3) the protocol to support the operations service and mapping to the underlying presentation service. This specification is targeted at database systems that support the Relational Model (SQL). This group initiated work on the ISO standard in late 1985 and additional development is needed. Through the CALS effort, DoD could have a significant impact on the content of the future standard.

4.2.2. GRAPHICS AND THE IRDS

The CALS effort is heavily reliant upon graphics support, especially in the area of product data definition and CAD/CAE. With the data dictionary or IRDS being the overall knowledge base for knowing what data is available and where it is located, the CALS IRDS must support graphics. There needs to be a specification for an IRDS optional module to support graphics data types and an interface to provide data definitions to a standard graphics system.

4.2.3. DATA MODELING AND THE IRDS

CALS is currently using data modeling methodologies to support the CAD/CAM area. For example, the Air Force's IDS is using IDEF1. The NBS AMRF is using another technique called SAM*. Specifications for an IRDS optional module is needed to provide appropriate analyses and reports for CALS data modeling.

4.3. CALS PROGRAMS AND STANDARDS

Each of the CALS programs have associated with it data which must be processed. This data is categorized into a functional view and one or more of the technical views. There are standards that may be appropriate for the different CALS programs. Each CALS

program that was reviewed during the two trips is briefly described below along with identification of the functional and technical view most appropriate for that program. The descriptions of the goals and objectives came from the implementation plans of each respective service. The NBS comments are general notes. This section will be expanded in the final report after additional technical review and analysis.

Project Title: Automated Technical Order System (ATOS)

Goals and Objectives: ATOS is being implemented to improve generation, storage, and distribution time associated with AF TO's. A major goal is to significantly reduce the cost of AF Technical Order acquisition and management.

Functional View: Technical Support

Technical View: Image and Text

NBS Comments: There probably will be a need for an index to the ATOS data. If so, either IRDS and/or SQL standards should be appropriate. Additional information and analysis is needed to determine the extent to which a part of the Technical Order and Change Order data is "structured" versus textual data. If part of the data is structured, then the NDL or SQL standards should be appropriate.

Project Title: Engineering Data Computer Assisted Retrieval System (EDCARS)

Goals and Objectives: The purpose of EDCARS is to make significant improvements in the quality, retrievability, and cost of Engineering Data.

Functional View: Product Definition Data

Technical View: Image

NBS Comments: The IBM's IMS database management system is being used to develop a "central index" or directory to the 1.5 million drawings stored on optical disk storage. For compatibility across the services, either the IRDS or SQL/NDL standard should be considered.

Project Title: Integrated Design Support System (IDS)

Goals and Objectives: The IDS objective is to apply advanced information management technology to integrate engineering data into what will appear to the user to be a "single" data base. IDS, therefore, will mesh with the Integrated Computer Aided Manufacturing (ICAM) program, ATOS, EDCARS, UDB, IMIS and LIMSS.

Functional View: Integrated management view

Technical View: Image, Text, and Alphanumeric

NBS Comments: A database of commonly used parts is being used to create the illustrations for the technical orders. This database of drawing information could be indexed and managed by the IRDS, SQL, or possibly NDL.

Project Title: Integrated Maintenance Information System (IMIS)

Goals and Objectives: To promote effective aircraft maintenance by enhancing capabilities of base level maintenance technicians through tailoring information to support specific needs. IMIS will provide a single interface to all required information and will supplement on-aircraft diagnostics to provide full-fault isolation capabilities.

Functional View: Logistics Support

Technical View: Image, Text, and Alphanumeric

NBS Comments: The Air Force is reviewing the content and format of the Technical Orders. There are plans to have "automatic" cross-referencing/indexing to maintenance information. The IRDS or SQL standard should be appropriate to store and retrieve the results. More information is needed on Technical Order and maintenance data to determine if the IRDS or SQL standard is appropriate for these categories of data.

Project Title: Geometric Modeling Applications Interface (GMAP)

Goals and Objectives: The GMAP program will enable the digital communication of product definition data describing air-cooled jet engine turbine blades and disks throughout the entire product life cycle including engineering analyses, manufacturing, logistics, and depot support. GMAP objectives include: to create a definition and model of engine blade and disk product data; to survey the state-of-the-art geometric modeling systems and application; to define the minimum requirements of a geometric modeling system; and, to demonstrate the integration of a prime contractor with its vendors and Air Force Logistics depots.

Functional View: Product Definition Data

Technical View: Image

NBS Comments: The IRDS appears appropriate for the Conceptual Schema required by GMAP. More information is needed about the "information classes" of entities/features to determine if any of the IRDS, SQL, or NDL standards are appropriate. In any case, the IDEF1* ER model is being used and the IRDS would be appropriate for supporting that data modeling environment.

Project Title: Unified Data Base for Acquisition Logistics (UDB)

Goals and Objectives: The objective of the UDB is to develop, demonstrate, and evaluate a logistics information database system which will assist weapon system designers and logisticians in incorporating logistics considerations into the early design of weapon systems. The UDB will provide logisticians with a flexible, efficient database application system designed to ease the burden of documenting iterative LSA tasks.

Functional View: Logistics Support

Technical View: Alphanumeric

NBS Comments: The Air Force is currently using Cullinet's IDMS database management system and the IDD data dictionary. Therefore, the IRDS and either the NDL or SQL standard would be appropriate for compatibility and exchange of data with other systems. There is a need for a graphics interface to IRDS and the appropriate database standard.

Project Title: Product Definition Data Interface (PDDI)

Goals and Objectives: The PDDI program will establish a digital interface between engineering and manufacturing which replaces the engineering part drawing. PDDI objectives are: to evaluate the Initial Graphics Exchange Specification (IGES); to specify manufacturing information needs from engineering; to create a prototype interface to exchange product definition data between dissimilar systems in a factory, and to demonstrate the program in a production environment with in-house manufacturing systems and commercial CAD/CAM Systems. PDDI seeks to lower the costs of creating, managing and communicating part descriptions for aircraft systems by allowing the data to be transmitted in a standard, public domain format.

Functional View: Product Definition Data

Technical View: Image

NBS Comments: The IRDS appears appropriate for the Conceptual Schema required by PDDI. More information is

needed about the "information classes" of entities/features to determine if any of the IRDS, SQL, or NDL standards are appropriate. In any case, the IDEF1* ER model is being used and the IRDS would be appropriate for supporting the data modeling environment.

Project Title: Integrated Information Support System (IISS)

Goals and Objectives: To establish and operate a test bed to validate the concept of Integrated Applications supported by an Integrated Information Support System (IISS) in a heterogenous computer and data base environment. In addition, the project is using a set of interim standards and procedures to guide the enhanced design of the IISS. The set of requirements established from 1984 prototype and 1986 production implementation designs will be the basis for enhancements to the IISS. The test bed will serve as a technology transfer vehicle, training facility and a central development site. The test bed realizing the full benefits will also serve as a facility to assist the DoD Community in achieving these benefits faster and with less risk.

Functional View: Integrated Management View

Technical View: Image, Text, and Alphanumeric

NBS Comments: IRDS, SQL, and probably NDL are all applicable. General Dynamics is currently using 300 IMS databases on the shop floor. They are currently using a "sophisticated" data dictionary system with the IDEF1* model "sitting on top." The neutral data manipulation language is SQL based.

Project Title: ADP System for Technical Data Management and Engineering (MASTER)

Goals and Objectives: MASTER is a system to support repair parts procurement. MASTER provides an automated system for gathering, updating and storing technical information to be used for the development of requirements packages for repair parts acquisition. MASTER provides listings of item configuration documentation, i.e., specification lists, drawing lists, packaging data sheets, engineering change proposals (ECP's) etc., and its condition status. In addition, MASTER provides a code to relate the status of data contained on the listing to the type of procurement the technical data will or will not support without further review or documentation updates. Data for MASTER is derived from the Technical Data/Configuration Management System (TD/CMS), Budget Stratification System, manual input, and contractor provided part number data.

Functional View: Logistic Support

Technical View: Alphanumeric

NBS Comments: The data supporting this project is stored in computerized files. There is a requirement to maintain a part number to document and cross-reference parts listings. In the future, this system is scheduled to be redesigned. At that time, the IRDS, SQL, and NDL standards should be considered.

Project Title: Automated Publications Production System (APPS)

Goals and Objectives: Provides a Research and Development tool in automated publishing technology to interface contractors and MSC's, and utilize consolidated government data bases for increased publication production productivity.

Functional View: Technical Support

Technical View: Text and Image

NBS Comments: There probably will be a need for an index to technical data. The IRDS and/or SQL standard would be appropriate. Additional information and analysis is needed to determine the extent to which the technical data is structured versus text. NDL or SQL would be appropriate for any structured data.

Project Title: Digital Storage and Retrieval Engineering Data System (DSREDS)

Goals and Objectives: The DSREDS program will provide engineering drawings and related data in an easily used electronic form for developers/producers of Army materiel and for input to other systems needing this type of information. Goals include yearly cost savings of 21.5M; reduction in administrative lead time and drawing revision costs; standardized system with capacity to support through the 90's.

Functional View: Product Definition Data

Technical View: Image

NBS Comments: See EDCARS

Project Title: Electronic Maintenance Publications System (EMPS)

Goals and Objectives: The Electronic Maintenance Publications System is a program designed to evaluate the

concept of paperless technical publications by (1) obtaining RAM data on, and user response to, paperless technical publications through limited fielding of EMPS, (2) learning to prepare electronic technical publications and (3) collecting cost-effectiveness data.

Functional View: Logistics Support

Technical View: Image, Text, and Alphanumeric

NBS Comments: This system appears to require all three views of the technical data: Image, Text, and Alphanumeric. It definitely requires Image and Text data. There probably will be a need for an index to the technical data. If so, the IRDS and/or SQL standard would be appropriate. In addition, the technical data also contains information about part numbers. This information must be obtained or passed to other systems containing logistics support data. This data is structured and can be contained in a structured database. The SQL or NDL standard would be appropriate for this type of data.

Project Title: Personal Electronic Aid for Maintenance (PEAM)

Goals and Objectives: This is a joint effort with the Army to develop test and evaluate an authoring and electronic portable field maintenance system. The Navy project is focused on the extensive use of graphics displays as troubleshooting aides for use by the maintenance technician. The system is being designed to provide the technician with a checklist of maintenance actions and step-by-step procedures in combined text and graphics format.

Functional View: Logistics Support

Technical View: Image and Text

NBS Comments: The IRDS, SQL, or NDL standard could be used in conjunction with the authorizing system to facilitate the associated logistics functions. Additionally, SQL may be an applicable tool for use by the maintenance technicians, even though a portable computer would not be large enough to support a full function DBMS.

Project Title: Computer-Based Aide for Troubleshooting (CBAT)

Goals and Objectives: The increasing sophistication of modern Navy weapon systems has resulted in an exponential growth in the technical information required for support. This phenomenal growth often does not include the additional documentation required to support maintenance of the advanced electronic circuitry beyond the point where the

automatic test equipment (ATE) and built-in-test (BIT) leave off. Complete reliance on ATE and BIT forces the technician, when ATE and BIT fail, to fault isolate without any additional technical information. While the amount of data is, in itself, a problem, complexity in the presentation of information aggravates it. Specifically, it is the user's access to an interaction with the technical information that is formidable.

The objective of this project is to define, describe, and evaluate the technical information variables that contribute to effective maintenance performance by: (1) development of a technology base for technical information design and delivery, and (2) design, development, test, and implementation of an intelligent user defined technical information system.

Functional View: Logistics Support

Technical View: Image and Alphanumeric

NBS Comments: CBAT is similar to PEAM insofar as it needs support for both authoring and electronic delivery of information. The IRDS, SQL, and/or NDL should be considered.

Project Title: Navy Integrated CAD-CAM

Goals and Objectives: This project will establish hardware and software system specifications, develop program documentation and execute consolidated acquisition and integrated installation of CAD-CAM systems at principal engineering and industrial activities performing design/maintenance of ships, systems and facilities.

Functional View: Product Definition Data

Technical View: Image

NBS Comments: The indexes which accompany the CAD/CAM drawings are not automated. The use of a DBMS to manage these indexes would be a step in the right direction; however, a better solution would be a DBMS integrated with the CAD-CAM software, so that selected text embedded in the drawings is automatically indexed, and so that all text is accessible via full-text search. The IRDS, SQL and NDL standards should be reviewed for applicability.

Project Title: Computer-Aided Integrated Logistic Life Cycle Support (Computer Aided Logistical Support Analysis) (CALSA)

Goals and Objectives: Apply RAM across the spectrum of logistic support analysis and the logistical support analysis record over the life cycle of the acquisition. To provide front end and supportability inputs and considerations to the design and engineering process for the acquisition process.

Using the government owned CALSA (with the MK 50 Lightweight Torpedo Program as a testbed), the first steps have been to record with current primary emphasis on the timely spare provisioning process, reviews and audits.

Follow-on efforts will integrate RAM/CAD on an interactive basis utilizing existent databases for corporate memory to provide front end inputs to the design and engineering process.

Functional View: Logistics Support

Technical View: Image and Alphanumeric

NBS Comments: The CALSA project should consider using an IRDS to support a user community encyclopedia of acronyms and definitions and to integrate access to a database of logistical support analysis records. SQL should be considered for most of the ad hoc reporting requirements.

Project Title: Engineering Drawing Automated Storage and Retrieval Equipment (EDASRE)

Goals and Objectives: The DLA Engineering Drawing Automated Storage and Retrieval Equipment (EDASRE) project is directed toward automating four technical data repositories to support information requests and reprocurement actions. The repositories include the Defense Electronics Supply Center (DESC), Defense General Supply Center (DGSC), Defense Industrial Supply Center (DISC), and Defense Construction Supply Center (DCSC). Repository functions of each of the Supply Centers are established under DoDI 5010.12, Technical Data Management Program. The DLA EDASRE program is comprised of two phases. The first phase provides DLA with an interim capability to fully automate processing of aperture card files at the four technical data repositories, thus eliminating the need to manually store and retrieve aperture cards in response to customer requests for technical information.

Phase 2 of the EDASRE program will incorporate DLA planning to transition from Phase 1 interim automated aperture card systems into DSREDS/EDCARS digital storage and retrieval environment for the processing of engineering drawings and reprocurement bidset packages. It is the DLA's intent upon

completion of the EDASRE Phase 2 planning to acquire digital capability through any existing DoD contract that may be used as a means for acquisition. Otherwise, DLA will look toward the DSRED/EDCARS experience in defining digital processing requirements and acquisition criteria before taking any competitive procurement action.

Functional View: Product Definition Data

Technical View: Image

NBS Comments: In planning for automating the drawing repositories, there is a need to develop a "central index" or directory to all of the engineering drawings. The IRDS or SQL/NDL standard should be considered.

Project Title: Modernized Parts Control Automated Support System (MPCASS)

Goals and Objectives: The Defense Logistics Agency Parts Control Automated Support System (PCASS) currently has many manual functions involved in the administration and performance of the DoD Parts Control Program, DoDI 4120.19, MIL-STD 965. This program supports Military Parts Control Advisory Group (MPCAG) operations. The MPCAGs are located at four different Defense Supply Centers (DSCs), which evaluate and make recommendations on parts proposed for use in systems being developed by the DoD and other Federal agencies. The MPCAG's responsibilities are to promote standardization through the use of military specification parts.

Objectives:

1. Provide an on-line database to replace PCASS sequential tape files, allowing near-real-time processing and ad hoc query capability for the military services and industry.

2. Provide for almost 100% growth to 1,000 contracts supported per year, with improved response time.

3. Automate the remaining manual/paper reference files, including status of Mil/Spec standard preparation (over 1,200 per year) and Qualified Products Lists (over 200 lists).

4. Provide telecommunications for system input/output, and for interface with reference files in other DoD and industry ADP systems.

Functional View: Logistics Support

Technical View: Alphanumeric

NBS Comments: The IRDS and either NDL or SQL would be appropriate to enhance compatibility and exchange of data with other systems.

5. STRATEGY FOR USE OF A DATA DICTIONARY

The Data Dictionary or Information Resource Dictionary System (IRDS) will be a valuable support tool for CALS. The benefits of an IRDS and its use in CALS are discussed in the following paragraphs.

5.1. IRDS BENEFITS

A preliminary cost-benefit overview prepared for ICST estimates that the U. S. Federal Government could realize over $120 million (in constant 1983 dollars) in benefits by the early 1990's from use of a standard IRDS. Opportunities identified for cost reduction and avoidance included the following:

Improved identification of existing, valuable information resources that can be used by others in the same organization or shared with other organizations

Reductions of unnecessary development of computer programs when suitable programs already exist

Simplified software and data conversion through the provision of consistent documentation

Increased portability of acquired skills resulting in reduced personnel training costs

5.2. USE OF IRDS IN CALS

In the arena of information systems management for CALS, the IRDS can be used in the planning process, the design and implementation of information systems, the integration effort, and the assessment of impact of change.

5.2.1. PLANNING PROCESS

The IRDS can be used in the planning process for documenting and tracking CALS systems and associated funding requirements, contract awards, and implementation status. The IRDS can serve as the focal point for basic knowledge about all the CALS systems. It can contain some of the basic funding requirements about information systems, although more specific funding information may be contained within the Planning, Programming, Budgeting System (PPBS). The IRDS can interface with the PPBS for more specific budget information. Likewise, general

information about contract awards can be maintained in the IRDS and the IRDS can interface with other systems to obtain more specific contract information. The information on implementation schedules can be maintained in the same fashion. More specific schedules would be obtained from a project management system.

5.2.2. DESIGN AND IMPLEMENTATION OF SYSTEMS

The IRDS can be used during the analysis, design, development, implementation, operation, and maintenance of CALS information systems. Each of these areas is discussed below.

During the analysis phase, the IRDS can be used to analyze and document the overall information needs and the data requirements. The strategies of CALS management can be translated into logistics functions that are needed and then into actual application systems. The data requirements can be translated into data models to support the logistics functions.

During the design phase, the logistics functions can be further defined in terms of more detailed and specific events that must occur. The data models can then be expanded to include the data elements required to support a CALS system.

During the development phase, the IRDS can be used to support the development of screens, reports, and other inputs, programs, and outputs. The physical database can be created from the previously created data model.

At implementation time, the IRDS can be used to integrate the installation of a specific CALS system with other CALS systems.

Once systems become operational, it is possible that the IRDS can be used in an active capacity. Some commercially available data dictionaries actively interface with other software, resulting in an integrated environment. In the long run, this is a desirable objective to pursue.

The IRDS can also be used as a tool for the maintenance of CALS systems. This is discussed further in a later section on impact of change.

5.2.3. LOGICAL INTEGRATION TOOL

The IRDS can serve as a logical integration tool for integrating life cycle, configuration, and project management functions related to CALS information systems. The IRDS can be used to verify that specific items completed in one life cycle phase are adequately documented and consistent before moving into the next

phase. In addition, the IRDS can aid the management of information systems configuration to ensure that systems are configured properly to accomplish established goals and objectives. The IRDS can also assist information system project managers in integrating their projects with other projects.

5.2.4. IMPACT OF CHANGE TOOL

The IRDS can also be used to aid management in assessing the impact of change to information systems. An IRDS containing all of the interrelationships of data and how it is used in the information systems can provide a detailed analysis on the impact a proposed change will have on any or all of the information systems. For example, the IRDS could be queried to determine the impact of changing a zip code from five characters to nine characters or the length of a part number. The IRDS would report on the programs and databases that would have to be changed. This capability can improve considerably an organization's ability to estimate the time and cost of proposed systems development or change. It will more thoroughly ensure that a given change to an item will be changed in all programs and systems using that item.

5.3. ARCHITECTURE OF MULTI-LEVEL IRDS

In the strategy for implementing the IRDS in CALS, there must be an architecture of multi-level IRDSs. The size of the CALS effort is too large to expect a single IRDS to contain all the CALS information resources. Therefore, there must be a multi-tiered IRDS approach. Each tier would contain the information appropriate for a specific level. For example, OSD might have an IRDS that contains global CALS information; each service could have a service CALS IRDS containing information global to the respective service; and then at the lowest level there could be an IRDS that would be an integral part of a specific CALS information system.

5.4. IRDS STATUS

As specified in an earlier section, the IRDS is a draft standard specification for a data dictionary system. NBS expects that the IRDS will be an American National and Federal Information Processing Standard by early 1987. Until the IRDS becomes an approved standard, CALS may want to start with a commercially available system for CALS management purposes and then migrate to IRDS standard implementations. It is likely that computer tools will be available to automatically perform this migration.

6. RECOMMENDATIONS

To ensure that standards used in CALS are appropriate and available when required, NBS recommends the following actions be taken.

The CALS Working Group on Standards and Specifications work with NBS to select a limited number of "representative" programs representing the three major categories of data: Image, Text, and Alphanumeric. These programs should also represent the three functional areas: product data, technical support data and logistics support data. The goal is to select three to six programs for in-depth analysis. NBS then can review other CALS programs in a more general manner to determine if the conclusions from the in-depth analysis are accurate for CALS in general. Specific technical issues that NBS must address appear in Section 2.4, Physical Characteristics. Specifically, NBS must perform more analysis on:

> Existing and planned databases to determine which of the database standards, SQL or NDL, are most appropriate.
>
> The indexes/directories in CALS systems to choose between SQL and IRDS as the preferred standard.

The DDF and ASN.1 standards be considered as possible alternatives for the interchange of data commonly contained within structured files or databases. For more information, see Section 4.1, DDF/ASN.1.

NBS complete the assessment of CALS needs and expand this report in the following areas:

> The data modeling methodology and tools required to support CALS (Sections 3.2 and 3.3) and the need for IRDS to support data modeling (Section 4.2.3).
>
> The requirement for distributed DBMS and IRDS to all CALS users access to the widely dispersed CALS data (Section 4.2.1).
>
> The support a DBMS in a CALS environment needs from an IRDS (Section 4.2)
>
> The support an IRDS should provide for graphics standards (Section 4.2.2)
>
> Strategy for using a data dictionary, or IRDS (Section 5).

DoD/CALS and NBS determine the need for an IRDS User Manual. Such a document would address the types of data that users need in an IRDS to: provide the necessary knowledge to obtain access

to CALS data; plan CALS information processing activities; assess the impact of proposed changes and subsequently manage required changes; determine problems that can occur if certain features of the standard are misused; etc.

DoD/CALS review Section 4.2, Research and Development Issues, in preparation for discussions of priorities with NBS.

CONCEPT PAPER

CALS REPRESENTATIVE SYSTEMS[1]
CONCEPT PAPER

1. In its initial quarterly report, NBS recommended that DOD work with NBS to select a limited number of "representative" programs which would represent the overall CALS initiative. These programs should represent the major components or application areas comprising CALS. Further in-depth analysis could be performed on these programs with the output being a generic model describing each application area. Each generic model would identify the common "core" characteristics and requirements of the Services, DLA, and industry, and would provide NBS with the information needed to identify standards that would apply, or to enhance or tailor those already identified.

2. It is essential that a limited number of representative programs be selected for analysis because of the large number of CALS related program initiatives within the DOD components. There are 82 CALS programs that are described in the plans prepared by the Army, Navy, Air Force, and DLA. It would be both impractical and unnecessary for NBS to review all of them, particularly in light of the initial on-site visits and workshops conducted by NBS during FY 1986. NBS feels a better approach is to group the programs into categories representing major CALS application areas. Selecting one or two representative programs in each application area will simplify the analysis process. Generic models would then be developed for each application area, based on in-depth review of the selected programs, and the results incorporated into the CALS Framework and Development Plan. These models would also become test cases to validate the construct describing a CALS architecture.

3. One possible view of the CALS environment, based on our review of the Service/DLA implementation plans, consists of six major application areas, as depicted in Figure 1. Because there are overlaps among these application areas, as well as different perspectives, this view of the environment should be integrated with others that are emerging (e.g., the DoD CALS Implementation Plan) to produce a common description of the CALS environment. The six application areas are: 1) Product Definition Data; 2) Engineering Data Repository Systems; 3) Automated Authoring, Publishing, and Printing Systems; 4) Technical Data/Configuration Management; 5) Technical Information Delivery, Maintenance, and Diagnostic Aids; and 6) Maintenance Information Systems. Overlaps depicted by the shaded intersections represent the sharing of data and/or functionality.

4. Each of the six application areas is described in the following paragraphs.

---

[1] Note: This paper reflects changes suggested by OASD on Dec 1, 1986.

a. Product Definition Data is the totality of data elements which completely define a product for all applications over its expected life cycle. This includes the data about the engineering drawings and the semantic description of these drawings, such as tolerance levels, material composition, geometry, topology, attributes, and features necessary to completely define a component part or an assembly of parts. Any programs involving the Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), or Computer Integrated Manufacturing (CIM) are included in this application area.

b. Engineering Data Repository Systems provide the support for storing, retrieving, and transmitting digitized engineering data. The goal is to provide a cost-effective, high-technology system needed for handling and managing the engineering drawings and related information required by engineering and technical personnel.

c. Automated Authoring, Publishing, and Printing Systems provide the capability to create, publish and print technical material. Included in this application area is the capability to print upon demand, distribute technical material in paper form to the users, and process changes to the technical material. It does not include the delivery of the material to the maintenance personnel in electronic form because this is included in another application area (described in the next paragraph).

d. Technical Information Delivery, Maintenance, and Diagnostic Aid Systems will deliver technical documentation to maintenance technicians in electronic form. It is a "job aid" field maintenance system which uses graphics displays as trouble-shooting aides to provide the technician with a checklist of maintenance actions and the step-by-step procedures for diagnostic analysis of problems. This application area is treated separately from the prior one because of the additional considerations for on-line interaction and for interfacing with other logistics systems.

e. Technical Data/Configuration Management supports the integration and management of all the automated technical information within each services programs. The Army has a batch oriented TD/CMS system that can serve as a basis for determining the requirements in this area.

f. Maintenance Information Systems support the automation of parts supplies, maintenance accounting, repairables transaction and status accounting.

5. The view of the CALS environment described above should be integrated with others that are emerging to help develop and refine the common "core" requirements for building a CALS Framework. From this CALS Framework, NBS can then continue identifying the standards that will be required to support CALS.

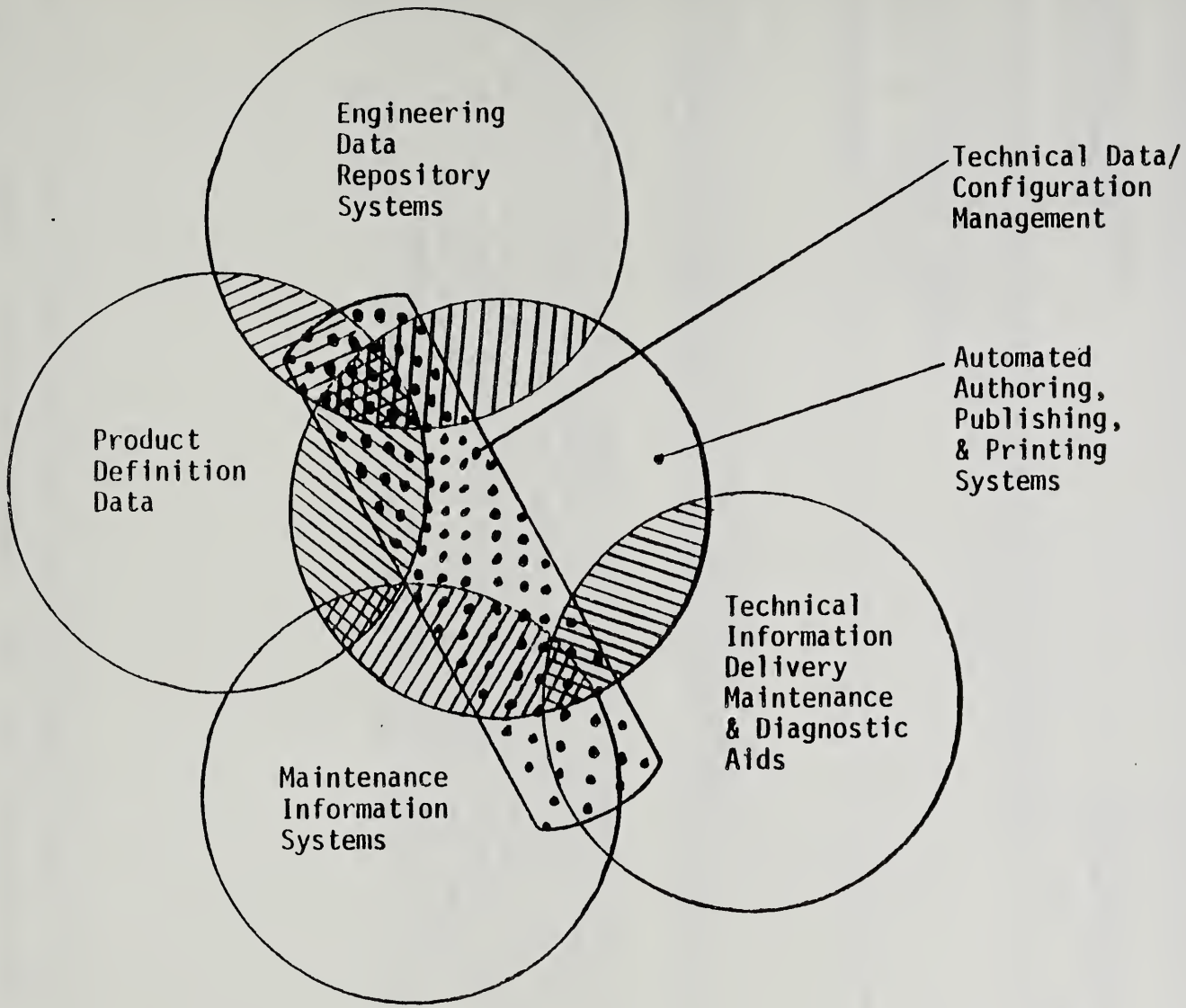


Figure 1 CALS ENVIRONMENT

NBS-114A (REV. 2-80)

U.S. DEPT. OF COMM. **BIBLIOGRAPHIC DATA SHEET** *(See instructions)*	1. PUBLICATION OR REPORT NO. NBSIR 87-3566	2. Performing Organ. Report No.	3. Publication Date

4. TITLE AND SUBTITLE

FINAL NBS REPORT FOR CALS, FY86

5. AUTHOR(S)

Edited by Sharon J. Kemmerer

6. PERFORMING ORGANIZATION *(If joint or other than NBS, see instructions)*

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

Final, FY86

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

Office of Assistant Secretary of Defense (A&L)/WSIG
Department of Defense, Pentagon
Washington, DC 20301-8000

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)*

The overall objective of the DoD Computer Aided Logistic Support (CALS) Program is to integrate the design, manufacturing, and logistic functions through the efficient application of computer and communications technology. DoD requires functional and interface standards and procedures that will enable the digital interchange of data in weapon system and automated system contracts, that will be common to all Services and DLA. This FY86 Final Report provides NBS recommendations for standards usage to support the interchange of CALS digitized technical information in four major areas: product data, graphics, database management, and text.

12. KEY WORDS *(Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)*

CALS; CGM; DoD; graphics; database management; IGES; IRDS; logistics; NDL; product data; SGML; SQL; text

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

15. Price

USCOMM-DC 6043-P80